2000-09

Evaluation of the Extensible Markup Language (XML) as a means for establishing interoperability between heterogeneous Department of Defense (DoD) databases

Hina, David Rex.

Monterey, California. Naval Postgraduate School

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

EVALUATION OF THE EXTENSIBLE MARKUP LANGUAGE (XML) AS A MEANS FOR ESTABLISHING INTEROPERABILITY BETWEEN HETEROGENEOUS DEPARTMENT OF DEFENSE (DOD) DATABASES

by

David Rex Hina

September 2000

Thesis Advisor: Valdis Berzins
Second Reader: Paul Young

Approved for public release; distribution is unlimited.

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	September 2000	Master's Thesis

4. TITLE AND SUBTITLE
EVALUATION OF THE EXTENSIBLE MARKUP LANGUAGE (XML) AS A MEANS FOR ESTABLISHING INTEROPERABILITY BETWEEN HETEROGENEOUS DEPARTMENT OF DEFENSE (DOD) DATABASES

5. FUNDING NUMBERS

6. AUTHOR(S)
Hina, David R.

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Postgraduate School
Monterey, CA 93943-5000

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES
The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(maximum 200 words)*

This thesis evaluates the application of current Extensible Markup Language (XML) tools and technologies toward solving data interoperability issues between legacy data repositories. Past efforts to address these issues have largely failed. XML has the capability to address many of the past problems, but this can only be accomplished when the supporting COTS tools and technologies are available.

The thesis first establishes the underlying issues that need to be addressed. It then evaluates the current state of technologies and COTS products and describes the advantages and disadvantages of each. Finally, it focuses in on the schema for a specific relational database, demonstrates a process by which data exchange can be implemented, and outlines the issues remaining to be solved.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Extensible Markup Language, Interoperability, Database Management	122
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFI- CATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

EVALUATION OF THE EXTENSIBLE MARKUP LANGUAGE (XML) AS A MEANS FOR ESTABLISHING INTEROPERABILITY BETWEEN HETEROGENEOUS DEPARTMENT OF DEFENSE (DOD) DATABASES

David Rex Hina
B.A., DePauw University, 1987

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN SOFTWARE ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
September 2000

Author: [redacted]

David Rex Hina

Approved by: [redacted]

Valdis Berzins, Thesis Advisor

[redacted]

Paul Young, Second Reader

[redacted]

for Luqi, Chairman
Software Engineering Curriculum

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

This thesis evaluates the application of current Extensible Markup Language (XML) tools and technologies toward solving data interoperability issues between legacy data repositories. Past efforts to address these issues have largely failed. XML has the capability to address many of the past problems, but this can only be accomplished when the supporting COTS tools and technologies are available.

The thesis first establishes the underlying issues that need to be addressed. It then evaluates the current state of technologies and COTS products and describes the advantages and disadvantages of each. Finally, it focuses in on the schema for a specific relational database, demonstrates a process by which data exchange can be implemented, and outlines the issues remaining to be solved.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEGEMENT

The author wishes to thank his wife Amy for her support and understanding during the many late nights required for this effort. He would also like to thank Dr. Valdis Berzins for his guidance and support.

I. INTRODUCTION

A. SUMMARY

The Joint Battlespace Infosphere (JBI) [JBM00] has been introduced as a concept that will allow the wide sharing of data between heterogeneous systems across multiple domains. The producers of the data, which under this concept remain largely unmodified, are the legacy Department of Defense (DOD) systems upon which battlespace operations depend, currently and for the foreseeable future. The ultimate consumers of the data include legacy applications, web access, commercial-off-the-shelf (COTS), and other mission-specific applications. The middleware layer that makes this level of data sharing possible is based on the eXtensible Markup Language (XML) and its associated specifications and technologies. It is the goal of this thesis to describe the software architectures and available COTS technologies that can be applied to bring this concept closer to reality.

There is currently a large amount of data within the DOD that is restricted to being utilized within either a single system or by a specific group or entity. This restriction is primarily caused by the differences between various software systems and databases. While mechanisms do exist for translating data between the different database systems, these mechanisms are typically restricted to a

single application and are easily affected by any changes that occur. Additionally, due to their custom nature, they are able to make little use of COTS tools for reducing the cost involved in development and upkeep.

XML is a recent technology that is ideally suited for taking data from diverse representations, and reconciling it into a common format that is portable and has a simple interface for data retrieval. It also has the advantage of being an open standard for which COTS products are constantly being developed and improved. A close look needs to be taken at how XML is currently being used to improve data interoperability, and at the future approach that should be taken, given the current state of the technology and existing tools.

B. RESEARCH QUESTIONS

This thesis will answer the following questions:

1. What are the issues that complicate data exchange between the systems of interest?
2. In what ways can XML and its related technologies offer solutions to these issues, and where are the deficiencies?
3. What are the XML specifications, technologies, and products that are applicable for database to database exchange of data?
4. What is the current state of these products and technologies?

5. What process can be followed to successfully apply these products and technologies?

The answers to a number of these questions will be presented within the context of the Global Command and Control System (GCCS) Integrated, Imagery and Intelligence (I^3). This system is representative of the legacy environments where data sharing is becoming a necessity. It will be used to illustrate the concepts laid out in the thesis and to demonstrate some of the issues that need to be addressed.

C. MOTIVATION

Traditionally it has been a difficult, time consuming, and expensive task to share similar data between different database systems. This has been largely due to the fact that there is no single standardized format for data transport between these heterogeneous systems, and no set of COTS tools to provide cost effective support. The primary benefits of this thesis will be to provide an analysis of where and how the use of XML, in its current state, can be applied to improving this level of data interoperability in a cost effective and timely manner.

D. ORGANIZATION

This thesis is organized into the following chapters:

- Chapter II provides background for the thesis and summarizes existing literature that is applicable to both data interoperability within the DOD and to the use of XML for data sharing purposes.

- Chapter III identifies the process by which XML can be used for structured data transport. This includes the challenges and considerations that need to be addressed in the use of XML.

- Chapter IV presents an analysis of relevant technologies and a look at the types of COTS tools that presently exist to support these technologies. Specific examples of COTS tools illustrate each of the tool categories.

- Chapter V presents and evaluates a process for applying XML technologies and tools to the GCCS-I3 database segments. This is done in the context of both Sybase Enterprise Server, the primary Database Management System for GCCS-I3, and middleware data translation and mapping tools.

- Chapter VI provides thesis conclusions and recommendations.

II. BACKGROUND

A. DOD DATA INTEROPERABILITY

1. Understanding the Issues

There are many barriers to data interoperability within the DOD and they have been well documented [NRC99]. When most information systems are first developed, they address a single, very specific set of requirements. The data formats are typically chosen to best suit the mission at hand, with little regard to standardization with other existing systems. When, after a few years, the need arises to communicate with another system, pairwise interfaces are developed between the two systems to support this need. This gives birth to a tangled web of directly interconnected systems that become increasingly difficult to maintain.

a) StovePipe systems

The problems associated with sharing data between systems are inherent in the original design of most legacy systems. When systems are designed and developed to operate in isolation, there is no motivation to consider the need to share data with other systems. Over time, this has changed as users have demanded access to multiple data sources from within a single application. With the advent of the Internet and with the greater emphasis placed on communication, the advantages to shared data and to the use of open data

formats are being realized, and the result is a change in the way systems are designed.

Unfortunately, within the DOD legacy data systems cannot just be redesigned from the ground up since they are critical to daily operations. A phased approach is required to continue the use of these systems while making the move to a shared data environment. Migrating legacy systems, however, is not an easy task. In evaluating the set of problems that must be addressed, Renner [REN96] focuses on the move to a shared schema, the extraction of knowledge encapsulated within the legacy applications, and the risk involved. He states that a shared schema is necessary, but it is difficult to achieve because it requires reverse engineering the schemas of the individual applications. He also points out that the existing applications contain valuable knowledge that cannot be easily discarded. Another area of concern is that a simultaneous cutover of all applications cannot be required, since the risk of failure with the operational systems is too high.

Due to the problems associated with moving directly to a shared schema, many legacy systems have taken the route of creating applications dedicated to providing the interface between each pair of directly connected systems. These applications contain the necessary knowledge to convert between the different schemas. As Renner and

Scarano [REN96] note, this method of communication is expensive because of the development and maintenance involved and because, with the pairwise interfaces involved, the cost increases with the square of the number of systems involved.

Another common approach to getting single use, stovepipe systems to play in this new world of shared data, without the move to a fully shared schema, has been a layered approach. In this approach data from different legacy data stores is maintained separately and with different schemas. Applications are modified to interpret data from each of the systems and provide displays on the same screen, but data from each individual system is layered one on top of the other. From the end user's aspect, these systems can be frustrating and difficult to use because of the lack of synchronization between the layers. They also lack scalability, since there is no true integration at the data level between the different layers.

One other approach that is more appropriate under some circumstances, and the approach discussed in this thesis, is to create a shared data server that accepts appropriate subsets of data in a central schema format from the external legacy systems. [REN96] states that this approach to integration involves three tasks: "developing a data model and data elements for the shared data, converting

the legacy data values to this new representation, and modifying the application programs to use the shared data server and its schema."

b) Diverse Data Representations

While many of the large data stores within the DOD are handled by very capable DBMSs, such as Oracle, Sybase, Informix and others, there are a number of issues that hinder interoperability between these systems.

One issue is the lack of a common vocabulary for use between the systems and the lack of a framework to support such a concept. While the need for a common vocabulary has been recognized for years, attempts to make it a reality have been met with very limited success. This has been largely due to the fact that there is often little motivation for data providers to spend the time and effort that it takes to integrate their individual data definitions into a central repository. Other tactics, such as forcing data providers into modifying existing systems to use a common set of terminology, also have not produced results.

One issue that hinders the move to a centralized vocabulary, as Hodges and Buck [HOD00] point out, is that the only way to truly understand the semantics of the data is to analyze both the structure of the data and the legacy applications that use it. This can be a costly process and very difficult to perform.

Another problem is the different types of data stores that exist in some of the legacy systems. While the majority of the data within the DOD exists within relational data stores, other formats that must be handled include both flat file and hierarchical structures. These data stores often do not have a standard mechanism for accessing the data, and may require complex, single use application programming interfaces (API) to extract, update, and delete data.

c) Time and Cost of Change

The cost of migrating these systems to exist in a shared data environment, for the reasons listed above, can be very high. History has shown that the timeframe required for the migration can also be unacceptably long.

[ROS00] points out that a large part of the cost historically has been faulty assumptions that are made in the approach taken to solve the integration problems. Some of these assumptions include a focus on the end task instead of an incremental, phased approach, insisting that all participants in the integration process adapt the same standard data models and data definitions, and that mandates are sufficient for getting all participants to contribute meaningful metadata about their systems. They also express concern that top-level infrastructure spending can be wasted, because it provides no motivation for the individual

systems to assist with the interfaces necessary for data integration.

2. Interoperability Strategies

a) Shared Information

Rosenthal et al. [ROS00] state that the goal of data integration is often portrayed as "all data available to anyone, any way they choose, anywhere, and at any time." The proponents of such a broad view typically include in their vision the following:

- Combinations of legacy systems, new systems, and data
- A universal centralized schema
- Metadata describing the individual systems
- Intelligent middleware to connect requesting applications with specific data sources.

This is followed by the observation that broad visions and goals such as these usually lead to failure.

Instead, they propose a more realistic view, with the recognition that constant change is inevitable, sources and consumers of data will be varied and unpredictable, and access to data is not sufficient, but true integration at the data level is required. This can only be achieved by treating data integration as a continuous process, and building in support for continuous improvements.

b) Metadata Tagging

Rosenthal, Sciore, and Renner [ROS97], in their paper "Toward a Unified Metadata for the Department of Defense," state that "data sharing within and among large organizations is possible only if adequate metadata is captured." The paper goes on to discuss the metadata requirements and approaches for metadata collection for a data sharing infrastructure. They state that the most difficult task is the collection of the metadata, and that this should be a collaborative effort between individual system builders, who possess the system knowledge, and the organization responsible for the overall interoperability effort.

c) Common Vocabularies

Another area that is a requirement for true information sharing is that there exist a framework for defining a common language between the different data sources. There are a number of approaches that have been taken to provide standardization, both in and outside of the DOD community. Some of the primary approaches will be described here.

ISO 11179 [ISO97] is a draft standard to establish such standardization in the form of a metadata registry on an international scale. The basis for this standard is to define data element classification schemes, and to use these

schemes to build and populate classification structures. In addition to classification, the standard specifies basic attributes that data elements should possess, rules and guidelines for data definitions, and specific naming and identification conventions.

While each of these are important considerations for a centralized registry, Rosenthal et al. [ROS97] point out some of the shortcomings in this approach, with the primary problem being the lack of specific features, such as actual schema specifications and APIs. Without these, the support of vendors and developers required for widespread adoption is less likely.

A DOD standard for data modeling, DOD 8320.1, was established in 1991 with the goal of collecting data element definitions across the DOD. It outlines a set of procedures for data element standardization, including the Defense Data Repository System (DDRS), which is a central database that includes data standards in terms of standard entities, data elements, and data models.

These past efforts to create centralized catalogs of metadata for the entire DOD or on an international scale have largely failed. This has occurred partly because the individual system developers have not had a real incentive to either pull from or contribute to these central repositories and partly due to maintenance issues with the

metadata. More recently, the Shared Data Engineering (SHADE) Team introduced a smaller catalog effort as part of the Defense Information Infrastructure (DII) Common Operating Environment (COE). The Joint Common Catalog (JCC) is not aimed at becoming a central catalog to handle all DOD metadata, but rather, as described in [HAS00], it will be "a set of components that can be used to create local catalogs as required that can interoperate because of their common features." One important part of the JCC is an XML Namespace Registry, which is an XML representation of the metadata elements maintained by the JCC. As a result of features in the XML specification, the registry will be able to provide runtime access to the metadata repository.

3. Joint Architecture Approaches

a) Joint Common Database

The Joint Common Database (JCDB) [CAR00] brings together in a single data architecture many of the components that this thesis will promote as being central to developing a true shared data environment. It is being developed as a single data repository that integrates data from multiple disparate sources, to generate a Common Tactical Picture.

The JCDB will combine data from multiple external inputs to develop its common data store, and then use a distributed approach to provide a reliable common data

source for applications needing it. Elements that are central to the JCDB's interoperability data model include a Joint Data Dictionary, a set of translators from legacy data stores, and the use of a standardized, semi-structured, data transport format in XML. Hayes et al. [HAS00] expand upon the need for a central data vocabulary as part of this effort and discuss the current state of the JCC. The usefulness of data translation layers and semi-structured data formats is discussed later in this chapter.

b) Garlic Fries

One of the areas where there has traditionally been a lack of interoperability has been between dynamic, near real time (NRT) data sets and less volatile relational data sets. Specifically, within the GCCS-I3 Common Operational Picture (COP) environment, this problem exists between the relational data stores such as the Modernized Integrated Database (MIDB) and the NRT data maintained by the Tactical Management Service (TMS). Track data managed by TMS, which is usually very time sensitive, lacks any consistent method for correlation with similar data within the relational data stores. When new tracks are established within TMS, due to the differences in data representation and the dynamic nature of some of the track parameters, new track IDs are established even when the object is already represented by an existing track ID within the MIDB. One

result of this synchronization problem has been the lack of ability to maintain a longer term history of tracks in TMS.

Garlic Fries is a system currently being developed to address both the synchronization and archiving problems. Synchronization between the data stores is provided by a set of translation filters, correlation logic for associating the NRT tracks and the more static relational data, and a new data store to handle cross-references between the two. Once the data from both data stores is in a common format, the correlation logic will utilize timestamps, positional information, and other attributes to correlate the track representations. Archiving of the TMS tracks will be handled using XML for data transmission from TMS and the relational data stores, and for use by other mission applications. This will allow the client applications to make use of an extensible, standardized interface that is abstracted from the underlying data structures. [FGM98]

This is an example of how XML is finding its way into a DOD data interchange environment. It also points out one of the advantages of using a semi-structured data format such as XML, which is that messages in a single format can function as both a medium for structured data transmission and for formatted data display.

B. USING XML FOR DATA INTERCHANGE

1. Why XML?

In it's most basic sense, XML is a method for transmitting and storing structured information. It allows us to apply meaning to information, in such a way that the information can then be indexed, searched, displayed, and manipulated with greater ease than it might otherwise have been.

The XML Specification [XML98] states that "XML describes a class of data objects called XML documents and partially describes the behavior of computer programs which process them." In the annotated version of the specification, Tim Bray further states that an XML document can be represented in a number of different ways, including as a file, a record in a relational database, an object delivered by an object brokering system, or as a stream of bytes at a network socket [BRA98].

2. Characteristics

XML is a mechanism for describing content, and as such, it has a set of characteristics by which it provides this data description. Some of these characteristics are important to consider when comparing XML to other mechanisms for structured data storage, such as relational and object databases.

XML, as it's name implies, is extensible. One of it's greatest strengths is that it allows a language to be created that is specialized for a particular area of use. Within this area of use, a document type definition (DTD) can be defined that specifies the set of tags to be utilized for documents pertaining to this area. The DTD, in addition to defining a grammar, can also, to a limited extent, function as a schema for the document. Specifically, the XML specification does give the DTD the ability to express which data elements must be present, what attributes they must have, specific ordering of the data elements, null data constraints, and limited functionality for expressing data element uniqueness. The DTD lacks the capability, however, to constrain the data type of elements, to express the allowable size of the data within an element, and to limit the allowable set of values an element can possess.

The structure of an XML document is hierarchical, and can always be described by a tree-like graph. This hierarchical structure is good for providing a clean organization of the data and for easy translation into other environments that require structured data. It also lends itself to other functions that commonly need to be performed on data, such as querying, searching and indexing.

Another characteristic of XML includes a method, through use of the XML *ID/IDREF* attributes, for associating

unique identifiers with each element, the basic unit of an XML document. This becomes important when translating data from an XML document to a structure that might require unique identifiers, such as a relational database.

To illustrate this concept a simple example follows. It consists of a list of two tracks, each of which has a globally unique track ID, a timestamp, and a location identifier. In a relational database, each track would be expressed as a single row in the TRACK table, with track_id representing the unique primary key. The DTD syntax for the TRACK element could be expressed as

```
<!ELEMENT Track (#PCDATA)>
<!ATTLIST Track track_id ID #REQUIRED
timestamp CDATA #REQUIRED
coordinates CDATA #REQUIRED>.
```

A corresponding XML document might be

```
<Track_list>
  <Track track_id="1001"
         timestamp="17203055"
         coordinates="325377680N1171033970E" />
  <Track track_id="1002"
         timestamp="17253000"
         coordinates="325325440N117102325E" />
</Track_list>.
```

In this example the use of the *ID* attribute guarantees that if the value for the track_id attribute is not unique, validation of the XML document will fail.

One characteristic of XML that is clearly an advantage over relational systems is the ease with which changes can be made to the data structure. Because structural information is maintained as part of the data itself, these changes can also be easily implemented independently of the source system. Using the previous example to demonstrate this characteristic, imagine that a system that processes the track data adds a requirement that in some cases the *coordinates* field needs to be expressed as two separate fields, *latitude* and *longitude*. This can easily be accomplished by adding a transformation filter at the point at which the data is received, so that when the necessary conditions exist, the resulting XML would become

```
<Track_list>
  <Track track_id="1001"
         timestamp="17203055"
         latitude="325377680N"
         longitude="1171033970E" />
  <Track track_id="1002"
         timestamp="17253000"
         latitude="325325440N"
         longitude="117102325E" />
</Track_list>.
```

This requires no changes to be made to the source schema, and the structure of the resulting data can easily be understood so that it can be processed in the appropriate manner. An equivalent change to a relational schema might require expensive and risky changes to be made to application code in many different locations, and the change to the structure of the resulting data can not easily be recognized by analyzing the data itself. This is an important difference between XML and relational systems.

3. Design Principles

One of the main reasons that HTML has been so successful as a display technology has been its simplicity. XML has been designed to provide the same level of simplicity, with greatly expanded functionality. The XML syntax is easily read and understood by both humans and machines. The options that the language presents have been kept to a minimum, making it much more unambiguous to work with from a developer's standpoint.

XML was designed to have the same features as HTML, but with added functionality and without some of the problems that HTML has experienced. In the same way that HTML has been successful in its ease of use over the Internet, XML was designed specifically for use in a widely distributed context.

One of the problems that HTML has had, however, has been the error tolerance required for applications that parse HTML documents. One of the most important aspects of the XML specification is the formality and precision with which it requires that conforming DTDs and XML document instances be written. As Bray writes, "Too many other standards and specifications have relied too heavily on prose and not enough on formalisms." [BRA98]

The XML specification introduces the concept of a *well-formed* document. This is an XML document that can always be unambiguously parsed to create a logical tree in memory, meaning that any parser should create the same tree structure. This allows a great deal of reduction in the amount of error handling that must exist in applications that process XML documents - they are either "well-formed" or they are not, in which case they do not get processed. A large percentage of the code in today's web browsers is there just for error handling. This adds complexity and variability in behavior, thus the formal and concise design requirement for XML.

XML was also designed to be directly compatible with SGML. This meant that XML documents should conform to not only the XML specification, but also to the SGML reference [ISO86]. This goal was added to leverage the existing set of SGML parsers and applications. Therefore, any XML

document should be able to be processed without error by existing SGML applications.

4. Where does XML fall short?

a) XML is not a database management system

XML is a text markup system, and it was not designed specifically for database management. XML does not possess some database-like features in the same way that DBMSs do not possess markup-like ones. A typical database management system possesses not only the ability to store structured data, but also methods for querying, viewing, optimizing, and processing the data in ways that the data can be easily and efficiently utilized in many diverse applications. XML, in and of itself, does not possess these capabilities, although a number of additions and extensions have been made to the original specification that make this capability more of a possibility.

b) Not always the most efficient solution

One of the original design goals from the XML specification stated that "terseness in XML markup is of minimal importance." This underlines the fact that efficiency was not a top priority in the design of XML. The emphasis was instead on clarity, simplicity, and wide area of application. This is one of the reasons that XML, although it is a method for storing structured data, will

not by itself replace all the functionality of a relational database.

5. XML and Structured Data

In addition to its use for describing and storing text data, XML's primary purpose is to transmit structured data [ABI00]. The origin of this purpose is largely a result of the need to provide a mechanism for moving structured data over the Web, which is composed of a wide variety of different types of data sources.

One of the stated design principles behind XML was that it must support a wide variety of applications. More specifically, it was intended to be a vehicle for exchanging data between heterogeneous systems, and as such it can represent data from a wide range of origins in a common format [BOS99]. This is accomplished in XML largely by the way that the structure of the data is described by a formalized, standard mechanism, and this data description is always either maintained as part of the data itself or in a directly referenced description document.

The most important structural characteristic of an XML document is that it is hierarchical - the data is represented in a hierarchy of nested structures. The order of the elements within this hierarchy is important and must match the order outlined in the DTD, if one exists. Additionally, XML does provide for element identifiers that

are unique throughout an entire document and across all element types.

One of the unique features of XML data is its capability to retains its structure and its meaning despite multiple transformations. When information is pulled out of a database for a specific purpose, such as for display or to be stored in an alternate format for later use, it often loses both its structure in relation to other data and its meaning in other contexts. Data from an XML document, since it contains the description of the data as part of the data itself, can maintain much or all of its meaning, despite having undergone one or more transformations or transportation to a different context.

6. XML Messaging Solutions

Standardized messaging has been used for communication within the DOD for many years. The data formats used in this messaging have been implemented in different ways, but their primary purpose remains the same - to provide a mechanism for data interoperability. Examples of these standards include USMTF, TADIL, and CIX.

The success of these messaging standards at providing interoperability is a subject of debate. One of the main reasons cited for their limited success has been the expense involved in maintaining the standards in the face of changing priorities and advances in technology. The

architectures of the systems on which they are based have been described as being inflexible, because they require messaging formats to be known in advance. Any additions and changes that need to be made to the message formats typically involve long, costly trips through standardization committees and development cycles [ROS97].

These messaging standards do, however, have aspects that make them necessary, and even attractive, for continued use, now and in the near-term future. They are widely used for military information exchange both within the DOD and with partner nations around the world, and they are based on years of experience with the collection of information exchange requirements. This infrastructure can, therefore, be immediately utilized as both a vehicle for communication and for information collection, reducing the expense and time required to achieve true data interoperability.

The application of the XML format to these messaging solutions is one approach to making the move from inflexible legacy systems to achieving flexibility via the use of XML. Mapping existing messaging systems directly to XML can be done by developing a set of XML tags that correspond to the data fields within an existing message format, and then using either a DTD or XML-Schema document to describe the constraints implied by the legacy system. Messaging systems usually consist of a fairly simple, hierarchical structure,

that maps cleanly into corresponding XML structures. One advantage of this approach is the continued use of existing extraction and input interfaces to the database, while achieving the desired resulting data format.

This has led to efforts that leverage existing messaging infrastructures within the context of advantages provided by XML. The following section discusses two such efforts.

a) XML-MTF

The US Message Text Format (USMTF), which is widely used by the US and it's allies, has hundreds of classes of strongly typed message formats and thousands of standard data element definitions. The MTF system includes many specialized tools and technologies for the processing of the hierarchical MTF messages, including validating parsers, document creation and editing systems, a query language, and processing and delivery systems.

XML-MTF was developed as an initiative for the continued use of MTF by its large community of users, while reducing the cost and effort required for its maintenance. The expectation is that the use of COTS tools for processing XML and the standardization features of XML will ease the process of extending MTF and make it more interoperable across systems.

Much of the current effort to utilize XML for improving interoperability within the DOD has been focused upon the use of XML-MTF. Schneider [SCH00] describes ongoing XML-MTF efforts as part of the Joint Battle Management Integration (JBMI) Assessment, to which this thesis is contributing, as having the following attributes:

- Modernizes military information standards through commercial technologies
- Capitalizes on 20+ year investment in military information requirements
- Leverages industry standard XML format
- Defines a standard XML mapping for MTF messages
- Provides simple software tools to support XML-MTF implementation. [SCH00]

It is important to note that while these efforts are critical to establishing the use of XML for interoperability and for leveraging existing channels of communication, this is just the first step in establishing true interoperability. The use of existing message formats brings with it many of the problems that have limited the success of messaging in the past. The systems that utilize XML-MTF will still remain largely inflexible and limited to specific subsets of data. Overcoming these issues will probably require the use of centralized registries and true

database-to-database interaction, such as discussed later in this thesis.

b) CIX Messaging

Within the Global Command and Control System (GCCS) community, the GCCS Common Operational Picture (COP) Coalition Information Exchange (CIX) is used for the communication of information between different systems. One of the primary uses traditionally has been the use of Broadcast mode for disseminating track data. Besides limitations common to other messaging systems, such as inflexibility and maintenance issues, the use of GCCS-COP in conjunction with non-GCCS parties has been limited by the message format, Over The Horizon-Gold (OTH-G), which lacks support outside of the GCCS-COP user community. [INR00]

This has led to the update of CIX software to handle messaging in XML format. This is discussed and expanded upon in Chapter V.

C. SEMI-STRUCTURED DATA MODELS

Semistructured data is often explained as "schemaless" or "self-describing," terms that indicate that there is no separate description of the type or structure of data. This type of data contains the description of the data as part of the data itself, unlike highly structured data representations, such as most relational databases. Such

data is much more portable and free from many of the constraints that are typically associated with database representations, but it can also be more difficult to represent more complex relationships between the data in a semistructured representation.

In a more complete description, Florescu and Kossmann [FLO99] describe semi-structured data as having the following characteristics:

- The schema is not given in advance and may be implicit in the data,
- The schema is relatively large with respect to the size of data and may be changing frequently,
- The schema is descriptive rather than prescriptive (i.e. it describes the current state of the data, but violations of the schema are still tolerated)
- The data is not strongly typed (i.e. for different objects, the values of the same attribute may be of differing types). [FLO99]

One common example of semistructured data is a file system hierarchy, which is typically represented in such a way that meta-information in the data itself is used to describe each of the data structures. This is very analogous to the hierarchical structure of XML documents and the self-describing nature of XML elements.

XML is a form of semistructured data. It exists in a hierarchical structure with the markup within a document representing the data description. The XML Document Type

defines the class of document, and this document type is defined in a Document Type Definition (DTD), which specifies the structure of the document in terms of the attributes and elements that it is made up of and the order and relationships between them. One important difference between the DTD and a relational database schema, is that in practice, the DTD is used purely for validation purposes, and the actual structure of the data is maintained as part of the data itself. The XML specification even allows valid XML documents that do not have an associated DTD. This is in contrast to the data in a relational database, which cannot maintain meaningful structure without the externally applied schema.

D. CHAPTER SUMMARY

It has been the purpose of this chapter to outline the issues surrounding the data interoperability problem, and to describe how XML can address these issues. One of the primary problems in the past has been an inability to adapt to change. This is one of XML's greatest strengths and one of the things that make it a good fit. It is important to understand the issues as they currently exist as well as the capabilities and deficiencies of XML, since this is the context on which solutions proposed later in this document are based.

III. ACHIEVING INTEROPERABLE STRUCTURED DATA TRANSPORT VIA XML

A. THE ROLE OF XML IN DATA TRANSPORT

The flexibility and standards based nature of XML make it a good fit for solving many of the problems that currently surround the exchange of data between disparate databases. The original XML specification, however, is deficient in meeting some of the requirements for directly translating the traditional relational data model into XML structures. Many of these deficiencies are being addressed by the application of XML Schemas instead of the DTD mechanisms outlined in the original XML specification. There is, however, a need to take a snapshot of the state of existing technology to determine what is currently possible.

One of the questions that remains to be answered is how many of the current standards and specifications that surround XML can be applied directly to solving DOD data interoperability issues. This question is made more difficult by the fact that many of the technologies are still evolving and many of the tools that support the technologies are chasing a moving target. Therefore, some amount of risk is involved with any large investment of resources in many of these areas.

B. DATA STRUCTURE MAPPINGS

Techniques for mapping between relational database structures and XML have been extensively discussed in the literature [BOU99], [BUC00], [FLO99]. Although the actual methods for implementing the mapping vary, there are a number of similarities in each of the approaches. Most current approaches make a correlation between database table and column structures and XML elements, subelements and attributes. They then make their own extensions to the DTD to handle additional information that is not handled by the XML specification.

There are two primary approaches that are usually taken for creating a relational database to XML mapping. In the first approach, template driven mapping, the structure of the desired XML document is first laid out in a template, which is just a well-formed XML document with the exception that it contains a set of processing instructions that exist within special tags. The processing instructions typically consist of SQL statements which are replaced by query results when the document is processed by the data transfer middleware.

This type of mapping can be very flexible, since the resulting XML document can be formatted as desired prior to any processing. In this approach, the actual mapping between XML elements and database structures does not need

to be predefined. The mapping is done dynamically, based on the processing instructions embedded in the XML tags.

The primary limitation of this type of mapping is the capability of the processing instructions that are included in the template. If the instructions are straight SQL, they inherit the limitations that come with SQL. This limitation can be minimized, however, by providing support for programming constructs such as looping and conditional execution, and by allowing result sets to be input as parameters to follow-on instructions. This can, however, increase the complexity of the template. In some situations, the embedded instructions could also pose a security risk if proper safeguards are not maintained on the templates. Another consideration is that this approach is really only suitable for one way transfer of data from a relational database to XML. Another approach needs to be used for moving data in the other direction.

In the second approach, model-driven mapping, the mapping is clearly defined up front. A data model of some type is utilized to describe the structure for the XML document, and a mapping is then defined between the XML elements and attributes and the database structures. In one possible data model for this type of mapping, a tree is utilized to describe the relationships between the data members, where each inner-node represents an XML element or

attribute, and each leaf-node represents the non-element data values. This tree represents the resulting XML document, and a separate mapping is used to establish the correspondence between the nodes of the tree and individual database structures.

By using a specific uniform procedure for performing the mapping, a DTD can be easily and automatically generated from a relational schema. A typical procedure would be as follows:

1. For each table in the schema, use an XML element.
2. For each column in each table, create an attribute or a child element that is restricted to containing data only (no child elements).
3. Designate a set of fixed attributes that will contain the Meta-Data. These will preserve specific data constraints, including data type, size, precision, and primary key/foreign key relationships.
4. Develop a specific set of appropriate values for data type and size.
5. Utilize some method, such as XML's ID attribute, which can uniquely identify elements, to specify primary and foreign key relationships.

The distinction over whether to use attributes or child elements for displaying data-only entities is not clear and has been a subject of much discussion. In general, entities that might be considered as properties of the parent element are often expressed as attributes, with other entities expressed as elements. As an example of each representation, consider a *Track* element from the *Track_list*

illustrated in Chapter II. Representing the entities as attributes would result in

```
<Track track_id="1001"
    timestamp="17203055"
    coordinates="325377680N1171033970E" />.
```

Alternatively, if the entities were expressed as individual elements, the result would be

```
<Track>
  <track_id>1001</track_id>
  <timestamp>17203055</ timestamp>
  <coordinates>325377680N1171033970E</coordinates>
</Track>.
```

The primary advantage of the first representation is that the DTD allows greater control over restricting the value of each entity. Extensions to the DTD are required in order to constrain the values contained within data-only elements. In general, the second representation offers greater flexibility in terms of extension and reuse, since elements allow hierarchical structure and repetition that is not possible with attribute representations.

When generating XML from relational data, the question of whether to represent the transition XML data in a deeply nested hierarchical structure, or whether to retain the relational structure within the XML document is important

and there exist supporters on both sides. Liam Quin [QUI00] bluntly states that a hierarchical format in which relational semantics are removed "is not at all suitable for data archiving or for data transfer." The primary reason he cites for this is that once the data is placed in this format, it is no longer in normal form, resulting in data that is duplicated and difficult to maintain. This type of data representation requires the use of references to maintain relationships within the data wherever foreign key relationships exist.

A contrasting view comes from Rosenthal, Sciore, and Renner [ROS97], where they point out that the use of messaging in non-normal form is well-established within the DOD, the Electronic Data Interchange (EDI) community, and others. It is their belief that if this type of transfer structure were eliminated, "the impact on existing operations and legacy systems would be too traumatic." They do however state that current messaging solutions are very expensive to maintain and they inhibit system flexibility.

The ultimate answer to this question is largely dependent in this context on the consumer aspect of the XML. If the data is destined for a central repository with multiple separate sources, such as in a hub-and-spoke architecture, the differences in the source schemas may drive use of the use of a true hierarchical model. The real

advantage here is that the relationship between the data is expressed as part of the data itself instead of being externally expressed within the business rules. If, however, the data is being transitioned directly between two databases possessing similar schemas, maintaining the relational structure may be more appropriate.

C. GENERAL CONSIDERATIONS

1. Data Transformation Model

One of the design decisions that needs to be made up front is the number and type of transformations that will need to be made on the data. This will be dependent upon the number of different schemas involved, the similarities between the schemas, the event model that is used, and a number of other factors. This decision will impact not only the mechanism used to implement the transfer, but also the flexibility and the complexity of the solution.

One model might consist of direct database-to-database mappings. In this case, the number of transformations required is $x(x-1)/2$, where x is the number of databases involved. The data, as it exists in the intermediate XML format, does not need to be transformed multiple times into different XML representations since it is targeted for a single destination. Advantages of this model include simplicity and reduced chances of losing the structural and relational meaning of the data.

A more flexible model involves a single extraction of data, which can then be distributed to multiple different database systems. This implies the use of a single central XML format, which is then transformed one or more times into formats required for any of the target databases. The addition of each database in this scenario adds two transformations that must occur, resulting in a total of 2x transformations, where *x* represents the number of databases involved.

The first model is less flexible and extendable since it relies on a special XML format for each source to target database pair. The second model, however, requires just one database schema to XML mapping per database, meaning that additional databases can be added more easily and changes can be made to existing ones more readily. The ability to utilize the second model, however, is greatly dependent upon the variability in the schemas of the different databases.

The second model might make use of Extensible Style Language Transformations (XSLT), which is a standard that was created for mapping XML document structure into either an HTML document or into another XML document.

2. Business Rules

One of the more difficult problems that must be addressed is handling the differences in business rules across the data repositories. The term *business rules* in

this sense represents the practices and policies of the organization which are embedded in both database schemas and in the applications that manipulate the organizations data. This problem of understanding and dealing with differences in business rules is not specific to the use of XML and is certainly not new for the DOD. The initial, and possibly more difficult issue, is extracting the existing business rules. These can be difficult to define since they are typically buried within application code and are often very poorly documented. As discussed in Chapter II, costly and time expensive reverse engineering is usually the only way to accomplish this task.

3. XML Schema Format

A number of different formats have been proposed for representing schemas in XML format. There are a number of considerations that must go into this decision, especially when considering a schema that will involve sharing across many different domains with diverse needs in terms of data representations. This is discussed in greater detail in later chapters.

4. Message Flow

Following the extraction of data in XML format, the requirements for the type of message flow between the data stores must be addressed. This can be very important when considering the challenges involved in transporting data

between typical DOD data storage facilities which may be in various geographical locations, with restricted bandwidth and unreliable connectivity between them.

5. Event Model

Another important consideration is the type of event model that will effect the data transfers. At one end of the spectrum is a periodic dump of the entire set of data. This could be based on a simple timer and maintained completely independently of changes occurring at the source database. The big advantage here is simplicity; there is no need to tie into the event model of the DBMS itself, and the queries made against the source database will not dynamically change. The potential disadvantage is delay time, and possibly an increase in message traffic, depending on the period of the database dumps and the rate at which data in the database is updated.

At the other end of the spectrum would be an update-driven approach that would trigger a data transfer whenever an update occurs to any field of interest. This approach implies the use of some mechanism within the DBMS, such as a database trigger, that is activated on updates.

6. Loss of Metadata

When moving data either from a database to XML format or from XML to a database, there are a few important aspects that must be taken into account. One of these is the loss

of metadata, or data description, that can take place. When storing data in a database, some of the information pertaining to the physical structure of the data can be lost. This includes the entity definition and usage and encoding information for the data. One example of this loss can occur with the use of identifiers which establish relationships between the tables of a relational database. Since XML can represent relationships hierarchically, these identifiers might be discarded when extracting data as XML. As this example points out, moving data from the database to an XML stream, and then back to a database will often result in a change in the resulting data structure or content - even if the relationship between the data is acceptably maintained.

It is possible to keep all of the metadata intact, with the potential loss of some flexibility and an increase in complexity. For the requirements of this analysis, since the data will be moving in only one direction, from source database to the target database, it will be acceptable to allow some loss in relational and structural integrity of the data. A determination must be made and clearly delineated, however, as to what loss will be acceptable while still maintaining the necessary meaning of the data within the target environment. Abstract data types and object modeling should be able to contribute to a solution

of this issue. Through these techniques, information attributes are relevant if and only if they are observable via a public method.

7. Data Types

The XML specification does not provide direct support for data types. In particular, it does not enforce type constraints automatically, although this can be achieved by following conventions that encode the constraints and by adding external software to check the conventions. Part of the reason for this missing capability is that XML was designed to address a wide variety of applications, with very few constraints and minimal options in the specification. This is in contrast to the standard relational database model, where all data is strongly typed. In XML, with the exception of unparsed entities, all the data in an XML document is considered text.

There have been numerous techniques proposed for typing data within XML and a variety of different implementations exist. This is still an area that is not fully developed and the subject of much research activity [ABI00].

The basic decision that needs to be made is whether to extend the DTD or to use one of the existing XML Schema formats. An example of extensions to the DTD can be seen in Appendix B. An example of the use of an XML schema, XML-

Data Reduced (XDR), can be seen in Appendix C. These two examples are different representations of the same schema, and a sample XML document that could use either type of schema is provided as Appendix A.

A comparison can be made between the two approaches by taking a look at how one of the elements expresses its data constraints. Expressed using the DTD format in Appendix B, the *Target_Name* element is expressed as

```
<!ATTLIST Target_Name
     dtype NMTOKEN #FIXED "string"
     dsize NMTOKEN #FIXED "54"
>
<!ELEMENT Target_Name (#PCDATA)>.
```

Here, both *dtype* and *dsize* are fixed attributes that extend the DTD to provide data type and size constraints for each data element. Since they are extensions to the DTD standard, they require custom code for validation of these parameters. The same element expressed in XDR format is

```
<ElementType name="Target_Name" model="closed"
content="textOnly" dt:type="string"
dt:maxLength="54"/>.
```

In this case, each of the attributes are part of the XML-Data specification [LAY98], so any product that conforms to this specification should be able to properly validate XML documents based on this schema.

The choice of one approach over the other will primarily be based on the availability of COTS products that are able to interpret the respective schemas and the amount of custom code that must be written to perform validation. There are currently more products available for validating against the DTD, but this will change in the near future as the XML schema specifications become more solidified and they become more widely used.

The differences between the approach taken will determine the amount of structure and meaning that is either preserved or lost during the transition from XML to relational database. It will also determine the amount of custom code required to perform the mapping and the complexity and flexibility of the resulting data structure.

In moving data from highly structured relational databases to the semi-structure of XML documents, the concern is primarily with maintaining the meaning of the data through the transition, simplifying the procedure for performing the transition, and validating the data during its transition. Data typing within a database environment, however, serves the purposes of efficient storage, optimizing data queries, and classifying the structure of each element so that a common set of operations can be provided for each element type.

An important aspect of data typing is providing a mechanism for validation. Standard XML parsers provide validation of the document structure by using the DTD. This does not handle, however, the validation of specific data types. As part of the data transformation model, therefore, there must exist a validation routine for each data type that takes into account the element size, precision, allowable characters, and other properties of the data type. Alternatively, the syntactic structure could be specified in enough detail so that some of these properties will be guaranteed by the parser by making the grammar restrictive enough so only valid data can be represented. This can require putting most of the information in the tags as attributes.

8. Performance

Although there are a number of advantages to using XML as a data transport, assembling and disassembling data as XML documents adds overhead that can affect the overall performance of the data exchange process. This thesis does not address performance issues in detail, but it is an important consideration that can affect the tools, technologies, and methods of implementation.

D. XML QUERIES

One of the requirements for transitioning data between databases and XML documents, is the ability to perform

complex queries against the XML structure. The query requirements for processing and retrieving data from a linked hierarchical structure such as an XML document are very different than those for a relational database structure. While query languages are fairly well established for processing data in relational databases, the same is not true of query languages for XML documents. A number of different approaches have been proposed for creating a query language that will address the requirements inherent in interfacing with hierarchical document structures, but none of these have been adopted as part of the XML specification to date.

There is a Working Group within the W3C that is dedicated to the development of XML query functionality. The purpose of XML queries, as stated by the latest draft document from the Working Group, is "to produce a data model for XML documents, a set of query operators on that data model, and a query language based on these query operators." [XQL00] This will play a very important part in how data gets retrieved from XML documents in the future. It will allow ad hoc queries to be made against XML documents, or repositories of documents, in a wide range of environments and in a similar manner to the queries run against highly structured databases.

Several approaches for translating data between structured databases and XML's hierarchical structure have been discussed in the literature. David [DAV99] concentrated on the use of ANSI SQL's inner join operation to perform transformations of data between a database and an XML document. This can translate relational data from a database into a hierarchical structure, suitable for storage as XML. Bourret [BOU99] describes two different approaches for mapping between an XML document and a database. In the first, template-driven mapping, there exists no predefined mapping between document and database, but commands are embedded in templates that get processed by the data transfer middleware. In the second type of mapping, model-driven, a data model gets imposed on the structure of the XML document and this gets mapped to the structures in the database and vice versa.

E. CHAPTER SUMMARY

This chapter has explored the use of XML for transporting data to and from relational database systems. The greatest advantage of using XML in this role is its flexibility. This flexibility brings with it, however, a number of important considerations, each of which need to be evaluated prior to designing a solution.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. ANALYSIS OF EXISTING XML TECHNOLOGY AND COTS TOOLSETS

The intent of the previous chapters has been to set the stage for the discussion that follows. A description of XML and its advantages and disadvantages as a tool for interoperability is a necessary preface to any analysis of how and where XML can be applied to solving data interoperability issues. This chapter covers the how and where by looking at relevant technologies and tools that currently exist and that can be applied toward resolving the issues expressed early in this document. An important part of this will be a description of the status of these tools and technologies, since many of the XML-related specifications and standards are still in their early stages.

A. RELEVANT TECHNOLOGY

One of the original design principles behind XML was that it support a wide variety of applications. A number of other design principles were directed at minimizing the complexity of XML and making it easy to use [BRA98]. A consequence of this is that although XML can be applied as a solution in many different application domains, it cannot alone provide the entire solution for most of the problems within these domains. To address this, there have been a number of related technologies and specifications that have

been developed to meet the requirements in these areas. This section discusses many of those technologies that are relevant for the exchange of data between relational databases. While this list is far from the full spectrum of XML solutions, it is a representative subset that can be applied directly to our discussions here.

1. Extensible Style Language Transformations (XSLT)

Extensible Style Language Transformations (XSLT), which are part of the recently approved XSL specification, specify transformations that can be performed on XML documents. In particular, this mechanism takes one XML document and transforms it into another XML document based on the static mapping information contained in a style sheet. This can be for the purpose of display or, more interesting for our purposes, to convert data to different DTD or schema formats. An example of using XSLT to perform this type of conversion is given in the *Track_list* example in section II.B.2, where the *coordinates* element from one schema exists as separate *latitude* and *longitude* elements in another schema. By applying an XSLT transformation and using an XSL stylesheet to specify the mapping between the schemas, the transformation can easily be performed.

In data sharing applications, transformations via XSLT might be applied at one or more points during the data transition. One use would be to apply changes to XML data

so that it will conform to a desired data format. As an example, consider a data source that produces XML data tagged with data types that are similar, but different, from those required by the format of the consumer. By applying an XSL template, and running the data through a transformer, we can produce valid data for the consumer without modifying the data source. This can be a cost effective way to create standardizing software wrappers for legacy systems.

Another possible use is to utilize XSLT to reduce some of the data from a specific dataset, or to completely reorganize a source document. This use will be applied later to the production of valid messaging formats from larger sets of XML data pulled from a database.

XSLT currently has the status of a *Recommendation* to the W3C [XSL99] and as such, it is relatively stable. Numerous products conform to the specification and have been developed specifically for performing translations using XSLT. A comprehensive list of these products can be found at the W3C web site for XSLT [XSL99]. There are also a large number of products, such as the BizTalk products discussed below, that have a larger scope, but which use XSLT at their primary transformation engine.

2. Extensible Query Language (XQL)

This section briefly describes the Extensible Query Language [XQL98]. Because this research focuses more on

utilizing XML for data transport and interoperability, and less on its use as a mechanism for storage, we do not cover details here. XML queries do become important, however, when data from a relational database is truly exported as XML or in the case that the data is stored within a true XML database. The discussion here focuses on XQL since it is likely to become the predominant query language for XML.

XQL performs the same function for the hierarchical XML structure as SQL performs for a relational database, in that it permits data access and manipulation. It has also been designed to assist with integration of multiple XML data sources. Of interest here is the design requirement that calls for the ability to perform queries on streams of XML data for the purpose of filtering, in a similar manner to the usage of Unix filters. This could be used for either extracting data from the streams, or for transforming the data stream to compress it.

The XQL language is still in the early stages of development at the W3C, although it is being based on other query languages that reached some level of stability. Both the query requirements and the data model have been recently submitted as a *Working Draft* to the W3C. There are a number of products that possess some form of either an XQL processor or one that handles a variation of the language.

These products are likely to change, however, as the standards solidify.

3. Simple Object Access Protocol (SOAP)

The Simple Object Access Protocol (SOAP) [BOX00] is an open messaging architecture, designed to transmit data from sender to receiver on top of the Hypertext Transfer Protocol (HTTP). SOAP is typically used to combine messages to create a request/response pattern. The contents of SOAP messages can be of any format, although the expected use is one or more XML documents.

While SOAP is not necessarily related to interfacing with databases, it is mentioned here because it has features that make it very attractive for use in the communication between different systems. It uses HTTP as its underlying transport protocol, for which numerous reliable COTS products exist. It has encoding features that can assist with encoding messages in binary format prior to transmission, as well as handling multiple XML document instances to be combined into the same message.

4. XML Schemas

As explained in sections II.B.2, III.B, and III.C.7, the DTD portion of the XML specification falls short when it comes to fully describing the set of properties that come with strongly typed data. It also, therefore, does not make it easy to validate these properties of the data when they

exist. The result has been a complete lack of standardization among the various implementations that must represent strong typing within XML.

This has led to a number of attempts to define a standard to address these problems. These efforts have now coalesced into a single standardization group, called XML Schemas, which plans to replace the DTD altogether for the use in applications requiring these additional capabilities.

Although the future use of XML as a data interchange mechanism seems to lie with one of these new approaches, this is an area that is still evolving. Most of the existing COTS tools have little or no support for XML Schemas and require the use of a DTD, although this situation is quickly changing.

B. COTS TOOLS FOR DATA EXTRACTION AND TRANSLATION IN XML

The COTS tools that are of most interest for this evaluation fall into one of two categories: middleware or parts of specific Database Management Systems. The first category of tools are relevant to utilizing a standard method for interfacing with a database such as Open Database Connectivity (ODBC) [ODB95] or Java Database Connectivity (JDBC) [JDB00], and then performing the translation from data elements to XML using a middleware solution that handles the mapping. In certain cases, when the data can not be exported directly via ODBC/JDBC due to legacy or

security issues, some middleware solutions can still be used with success by using an existing API or some different method for extracting and updating the data.

The second category of tools involves utilizing the features of the different database management systems for accessing and storing data as XML. Although there are several types of databases in this category, including relational, object, and true XML databases, only relational databases will be considered here since they are used by the legacy systems that are of interest.

As might be expected, the tools evaluated in this section vary from small tools, targeted to address one specific area, to larger environments that possess a great deal of functionality across a wide spectrum. The tools evaluated here provide only a sampling of the tools that are currently available in each category. One of the tools discussed in greater depth, Sybase Application Server Enterprise (ASE) version 12, represents the category that is specific to a single Database Management solution. This DBMS was chosen since it can provide a solution for data exchange from GCCS-I3, which is covered in greater detail in Chapter V.

1. Middleware Tools

The term "middleware" covers a large category of tools. In this context, the term is used to refer to XML tools that

lie between the source and destination databases, and provide data translation, manipulation, and mapping services. Data translation is a basic requirement for achieving interoperability in an environment where the data sources are directly modified.

In some cases, the use of third party middleware tools that do not currently have XML capabilities are worth investigating for integrating data from different sources. This would be feasible if the data is already exported as XML from the database, or if another mechanism can be used for relational to XML mapping. This option is worth considering primarily due to the power and functionality that exists within this category of tools.

One example is *Data Joiner* [DAT00] from IBM. It can provide integration capabilities on a large scale, such as transparent SQL access and relational joins across multiple different DBMSs. It also provides comprehensive APIs for working with data that can not be exported via standard access, such as ODBC and JDBC.

Another comprehensive tool for performing translations between different types of database formats is Microsoft's *Data Transformation Services* [DTS00]. This tool can operate independently of the DBMS to perform translations between a large number of formats. It has both offline and online processing modes, and while it does not yet have XML

capability as part of the translation services, it can be used in conjunction with *Active Data Object* (ADO) technology to provide this capability.

One type of schema mapping and data translation tool is Microsoft's BizTalk products and specifications. BizTalk is a comprehensive set of products that provides a number of services at runtime, including document validation against a set of business rules, translation of data formats, schema transformation, document transportation, and tracking and logging capabilities. Its transformation and mapping capabilities are built on top of an XSLT engine, which can perform mappings between a number of different formats.

Similar to other middleware products, the BizTalk server provides the processing functionality which maps data to an XML stream based on a mapping provided by each organization. The schema used by the BizTalk framework is currently implemented in XML Data Reduced (XDR) [LAY98] and Microsoft has promised that the XML Schema standard will be used when it is finalized and released by the W3C. This can provide a significant advantage over the use of a DTD for specifying the schema and for document validation, since it will handle many of the data constraint concerns described previously in Chapter III.

One difference between BizTalk and some of the other middleware frameworks, is that BizTalk is designed as an

end-to-end product, and handles the transmission of XML streams from source to destination over a number of different possible transport protocols. In the typical scenario, both the data source and destination would have BizTalk server platforms. The application layer would contain the business rules specific to that organization, would be responsible for retrieving the data as well formed XML, and would format it for handling by the server. The server then validates the documents, and processes them for transmission to the destination BizTalk server. The destination then performs XSLT translations and mappings into the proper destination format.

This framework is attractive because it contains many of the elements that need to be in place for end-to-end data sharing. This includes data translation and mapping, transport, and a schema representation designed to easily map to a relational schema without loss of metadata. It is, like many of the XML products, still in flux and dependent upon related specifications to be finalized.

XML-DBMS is a model-driven, open source middleware solution for moving data between relational databases and XML [XDB00]. This tool is discussed here because it is representative of a number of the open source tools that are available, some of which have the same approach as this tool.

XML-DBMS has been implemented in both Java and Perl, and consists of an API to a set of packages that provide services for extracting data from a relational database into an XML format, and for taking data already in XML and inserting it into a relational database. The product uses ODBC and JDBC interface standards for accessing the data, so it can be used with Sybase, Oracle, SQL Server, or any other database server that has JDBC or ODBC drivers. In order to perform the mapping, an object view of the targeted XML format is developed. This object view is then mapped to the relational schema by taking the object properties, represented by XML elements and attributes, and linking them to specific columns in the database. This mapping is then performed at runtime whenever data is moved to or from the database. This method has the advantage of requiring no modification to the database.

2. XML Enabled Databases

A number of the vendors of larger database management systems, including Sybase, Oracle, Informix, IBM, and Microsoft, have been integrating XML capabilities directly into their respective database systems. Although each of the systems is implementing XML in different ways, and sometimes for different uses, there are many similarities in the functionality provided. This section lists some of the

ways that this XML functionality is being provided and can be utilized.

In general, the XML capabilities being integrated into databases fall into two broad categories: document-centric and data-centric. Document-centric capabilities to focus on the storage and access of documents, which can be characterized by an irregular structure and larger grained database representation. An example of this might be a product user's manual, which can be stored in its entirety, or in parts, as XML in the database. In contrast, data-centric refers to a more regular structure where each XML element has a corresponding data element within the relational database. While the lines between these two categories are not always clear, the focus here is primarily on data-centric XML functionality.

There are a number of advantages to using XML extensions to existing databases rather than third party middleware. In many cases, performance enhancements can be expected since better optimization can often be performed by the database vendors because they have access to the underlying structure of the database. Additionally, since data can often be stored or maintained in memory in XML format, instead of the tags and hierarchical relationships being established when the data is requested, gains in performance and architectural simplicity can result. An

example of this is the use of database *XML views*, which can be used to maintain a subset of data in the desired XML format. Queries are then executed against the view instead of requiring joins between the different database tables. This capability currently exists only in a small number of database systems.

Disadvantages to the use of database XML extensions include the fact that not all database systems currently have the same level of capability, and there are a number of differences in the ways that the capabilities are being implemented. So, while it might be possible to implement a third party mapping and translation capability in a similar manner across a set of differing systems, the effort and complexity of developing directly to each of the individual systems might be extensive. Much of the XML functionality in these systems is also currently either in beta form or still in the process of refinement, so each database system should be evaluated on a case-by-case basis.

Most major database systems now have some method available for exporting data as XML. Relational database systems that have integrated XML capability typically provide this capability in three different forms. One form allows XML-formatted documents to be generated from the individual data elements stored in the database. Another form involves extracting the data and structure from an XML

document for insertion into a database. The third form of functionality allows entire XML documents to be stored as a single entity within the database. This discussion deals only with the first two forms and their specific application in the Sybase Application Server Enterprise (ASE) [SYA00] database, although comparisons to implementations in other database management systems will be explored.

Within Sybase ASE Version 12, this capability is provided through use of Java tools and the Java interfaces to the DBMS. This means that in order to transfer data to or from the database as XML, the Sybase Java API must be invoked from custom code. The functionality of the API is limited to performing basic mapping functions and is similar to that of some of the middleware products discussed. Details on the process can be found in [SYB00].

Sybase ASE currently lacks some of the XML capability found in other large DBMSs, such as Oracle 8i, DB2, and SQL Server. Specifically, these other DBMSs provide additional transformations on query results through the use of XSLT. Additional capabilities, such a publishing views as XML and facilities for conducting XML queries are not currently available within Sybase ASE.

3. Additional Tools

a) Parsers

Virtually every tool that works with XML requires an XML parser. The parser is a software component that takes an XML document as input, reads and interprets the structure of the document, and then returns the result to the application for manipulation.

There are two basic types of parsers, one that produces a complete data tree as output and another that is event based. Parsers using the tree model typically produce an entire structure representing the document in memory prior to allowing any operations on the data. Once the document has been completely parsed, the resulting structure is passed on to the application either for direct use, or in the form of an API based on the Document Object Model (DOM) [DOM00].

With the event based parser, the application registers specific events that it is interested in, and it is then notified of these events as the parsing is taking place.

There are numerous parsers available, either for standalone use or integrated as part of another product. Many of the products listed already have integrated XML parsers, so individual products will not be discussed here.

b) XML Editors

An important part of any work with XML is an XML editor. While not specifically used for database work, a good XML editor can assist the developer greatly with producing valid XML documents for testing and performing certain types of transformations.

Only recently have XML editors approached the capability of editors found in other disciplines, but some of the work in this area has begun to redefine the role of an XML editor. The functionality available includes parsing, validating and editing not only XML, but also XSL, DTD, DCD, and other schema dialects. Another helpful function is the automatic generation of DTD or the various XML Schema dialects from XML source. Other capabilities include two-way translation between XML and tabular formats and XSL translations. These capabilities together provide more of the Integrated Development Environment (IDE) approach found with many of today's mature development environments.

c) Compression Technology

One area of concern with respect to the use of XML has been its verbosity. Like any textual markup language, XML contains a lot of redundant information and carries with it a great deal of overhead in terms of the metadata. This has brought about the need for compression technology to

reduce the size of the XML output. There are a number of products now available that address this need.

The wireless community has needed to address the same problem, and has developed a specification for handling XML compression in a standardized manner. The *Wireless Application Protocol Binary XML Content Format Specification* [WAP99] was developed specifically to reduce the transmission size of XML documents, in order to allow a more efficient transfer of data in XML format. The specification addresses low level details, such as byte-ordering and character encoding, that normally do not need to be handled by XML applications. This work could easily be leveraged to provide the same type of service for database to database transfer.

C. CHAPTER SUMMARY

There is a wide range of COTS tools and technologies for XML, many of which offer solutions for interfacing with relational database systems. Choosing the proper solution, in many cases, can be difficult because there can be a number of potential solutions, each which have different, but overlapping, functionality. This is clear when evaluating the various middleware tools and the XML-enabled database systems, which provide much of the same functionality. There are also a number of significant

variations in how they are integrated, so each must be closely evaluated.

V. XML TRANSPORT FROM GCCS-I3

A. OVERVIEW

Up to this point, we have discussed in general terms methods for applying XML and its related technologies to interoperable data interchange. Here, we will look at more of the specifics. This chapter focuses on the use of XML technologies and COTS products that have been previously discussed, and a process for applying them to one specific database architecture, GCCS-I3.

The first part of this Chapter outlines the steps involved. It includes a section that lays out the design goals for the process, a description of the steps involved in the process, and a description of the GCCS-I3 MIDB schema, which are used for illustration. Following this is a description of a data model and examples of an XML-to-relational-database mapping that could be used. The second part of the chapter applies some of the XML COTS tools and technologies to assist with the process. Finally, an assessment is made of the advantages and disadvantages to the data exchange process.

B. STEPS FOR END-TO-END DATA EXCHANGE

1. Discussion

Early efforts to utilize XML for data sharing within the DOD have included XML-MTF [MTF00] and XML-CIX [INR00]

messaging. While these efforts represent a good first step towards data interoperability through an increased use of COTS tools, they will still be subject to many of the limitations of the messaging standards on which they are based (see section II.B.6). Basic MTF and CIX messaging, as they currently exist, are also too restrictive to allow the level of data interchange necessary to address the needs of a Joint Battlespace Infosphere, as described in [JBM00]. XML will provide us the opportunity to easily expand upon these messaging standards, and when communicating with systems that require legacy messaging, transform the same data into a valid MTF or CIX message.

A comparison can be made here to work that has been done within the METCAST weather reporting system. The Weather Observation Definition Format (OMF) [OMF00] is a recent application of XML by SPAWAR PMW-185 to address shortcomings in *weather observation reports*. The reports are issued in a number of different messaging formats that are similar in nature to DOD tactical messaging formats. The set of problems that needed to be addressed can be summed up as a basic inability to extend the messages to provide additional information in cases in which it is needed. Some of the information needed for interpretation of the messages was maintained externally to the messages

themselves, presenting problems when this information was not available.

OMF was developed to provide annotations that would extend the existing weather reporting formats. In a similar manner, the process described and assessed below is designed to create a data sharing environment that is usable by existing military systems, yet extensible enough to accomplish interoperability objectives.

2. Design Goals

A number of the design goals that are necessary for this process are similar to those for making the move from a legacy message structure to XML, such as those expressed in [MTF00]. The main differences would be related to the emphasis here on database to database transfer and a lack of adherence to all details of a specific message format.

A basic set of design goals for our process follow:

- The data exchange process must handle multiple databases, each with a different schema, running under different DBMSs, and containing data, some of which has the same or similar semantics.
- The schemas for the existing databases cannot be modified.
- Where appropriate, data must be easily transformed between different message formats, including, but not limited to, USMTF, CIX, NATO Allied Data Publication Number 3 (AdatP-3), and Theater Ballistic Missile (TBM) track format.
- Operations that act on the XML schema must allow the schema to change over time.

- The XML schemas must be simple to construct using data element definitions derived from a central repository.

- Use of standardized technologies and COTS tools must be used wherever possible.

3. Process

The design of an architecture for transfer of data between the databases can be broken down into a series of steps, some of which, in practice, might be combined to form a single step. Here, they will be addressed as distinct steps in order to remain implementation independent.

The primary step of interest, since it will have the most effect on the XML toolset to be used, is to develop the data model and method for performing relational to XML mappings. This will be discussed in some level of detail in relation to the Modernized Integrated Database (MIDB). Other steps, some of which will be covered in lesser detail include the following:

- Establish a common vocabulary.

- Analyze legacy systems to determine the subset of data to be shared.

- Analyze legacy systems to obtain data structures, semantics, etc.

- Determine mechanism for accessing the data (e.g. stored procedures/triggers, special APIs, ad hoc queries, etc.)

- Determine an XML Data transformation engine

- Determine mechanism and protocol for data transport.
- Determine tools that can be used to assist with or provide the capabilities needed for each of the items above. This includes determining the amount of custom code that will be required and the amount of risk involved with specific tools and technologies.

While each of these steps are important to the overall process, the ones that will be covered here involve developing a data model, interfacing with the database, and providing a mechanism for communication. In doing so, the large number of choices that need to be made will be reduced to a smaller subset. The process will be greatly simplified here and many of the details will ultimately need to be filled in, but the purpose here is to lay some groundwork for how this task can be achieved using available COTS products and existing technologies. Figure 1 shows a view of the overall architecture.

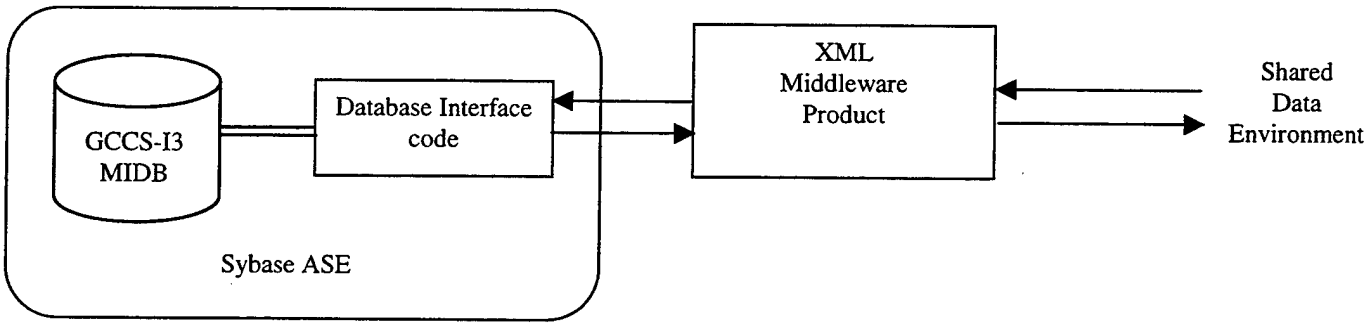

Figure 1: GCCS-I3 Data Sharing

4. GCCS-I3 MIDB Segment Schema

The Modernized Integrated Database (MIDB) serves as the primary data repository for general intelligence data within the DOD. There are a number of aspects about the MIDB schema, as it currently exists, that will affect the method used for extracting data from it.

One consideration is the version of Sybase ASE in use. The XML feature set provided by Sybase only exists in Sybase version 12, while the current version of the MIDB requires the use of Sybase 11. While middleware tools outside of the DBMS can be used to eliminate versioning problems, this decision needs to be made in advance.

The example set of data to be utilized references the following tables from the MIDB [MID98]:

- **TGT_MSN** - contains information about the missions against targets.
- **TGT_OBJ** - contains information for a military operation involving targeting.
- **TGT_LIST** - contains information on a prioritized, validated target set.
- **TGT_DTL** - refers to a specific target.
- **FAC** - contains data about a facility or an installation.

A subset of the elements from TGT_MSN are described below. The full descriptions of all elements can be found in [MID98].

MSN_ID	varchar(15), NULL
A unique identifier for the mission.	
OPERATION_NAME	varchar(54), NULL
The name used to describe an exercise or live set of military missions and activities.	
CLASS_LVL	char(1), NOT NULL
Highest classification level of the data contained within the record. Permissible values: U (Unclassified), C (Confidential), S (Secret), T (Top Secret)	
CODEWORD	char(1), NULL
Indicates the appropriate control channels associated with a physical records classification. Permissible values: 0 (collateral), 1-3 (SI-1), 4-7 (TK-1)	
MSN_NAME	varchar(30), NOT NULL
Name of the mission.	

The tables, elements, and constraints being used in the sample data are fully described in [MID98]. A full example in XML format is given in Appendix A. The corresponding DTD for the example is given in Appendix B. The same representation, but in XDR schema format, as used by BizTalk Framework, can be found in Appendix C.

5. Data Model and Mapping

One of the goals here is to provide a data model that will allow mapping to the common DOD messaging formats, but without the full spectrum of associated constraints. This will be similar in nature to the data model introduced for the *Observation Markup Format* introduced in [OMF00].

Examples will be given of the data model as a DTD and also as an XML Data Reduced (XDR) schema. Use of a DTD requires extensions to the XML Specification in order to represent data characteristics such as data type, repetition constraints, and other similar constraints. Some other characteristics, such as designating uniqueness or designating values as being required, are part of the specification.

Extending the DTD will mean that some of the data validation will not be performed by standard COTS tools, but additional checking will need to be performed in external custom code. While extensions are not necessary for any of the XML Schemas, such as XDR, the specifications for these standards have not yet been solidified, so the availability of tools is limited and subject to change.

The method used here to extend the DTD is to designate, for each element, a set of fixed attributes with values that describe the properties of the element. An example of this,

which describes a Mission_ID element as having a datatype of *string* with a maximum size of 15, is:

```
<!ELEMENT Mission_ID (#PCDATA) >

<!ATTLIST Mission_ID dtype NMTOKEN #FIXED 'string'
dsize NMTOKEN #FIXED '15' >.
```

This set of fixed attributes will be associated with every PCDATA (character-data-only) element. An additional set of attributes can be easily added to express other properties of an element that need to be checked. The datatypes utilized, for simplicity, are those from the XML-Data [LAY98] submission to the W3C. The full DTD listing can be found in Appendix B.

The example above, described using XDR, would be:

```
<ElementType name="Mission_ID" model="closed"
content="textOnly" dt:type="string"
dt:maxLength="15"/>.
```

Note that XDR is expressed in XML syntax, and the element description consists of an empty element with multiple attributes for describing the data properties. The attribute *model* refers to the ability to extend the element data description. The *content* attribute describes the content of the element as only text, only other elements, mixed, or empty. The *dt:type* attribute describes the data type. Additional information can be found in [LAY98] and the full XDR listing is in Appendix C. Both these

representations provide the same set of data constraints. The primary difference between the two, as discussed in section III.C.7, is the support provided by current COTS applications and the corresponding amount of custom code required.

In the relational to XML mapping, tables can be represented as elements which can only contain other elements. Columns can be represented as either PCDATA elements or elements that contain other elements, for the case in which the column expresses a relationship. The decision could have been made to represent columns as attributes, but representation as an element was chosen to provide greater flexibility.

It would also be possible here to express relationships within the data as primary key/foreign key fixed attributes. This would have the effect of flattening out the XML structure, providing a closer mapping to the relational structure and reducing data redundancy. The structure described, however, provides a more natural transition to messaging formats.

C. APPLICATION OF EXISTING COTS TOOLS AND TECHNOLOGIES

This section explains potential use of the various COTS product types for a number of the steps listed above in order to achieve the design goals listed. Additionally, it

covers the use of XML related technologies as they might be applied to assisting with this process.

As indicated by the previous discussion of the current state of XML technology and the challenges that exist when applying it to interfacing with database systems, there are a number of decisions that must be made in order to design an architecture that can support data exchange via XML. Since many of the specifications and tools that form the basis for XML are still in flux, there are many areas where choices need to be made. Most of these choices are dependent on the existing architectures of the systems involved.

1. Analyze Data Structures and Semantics

The goal of this step is to understand data structures, relationships, and rules that apply to the data well enough to map the data to a different schema without a loss in meaning. Much of this work must be done manually. The level of effort required may depend on a number of things, including amount and quality of documentation, complexity of the data relationships, type of data storage, and amount of business knowledge that is buried in applications.

One concern with the MIDB is the use of "tie" tables throughout the database for establishing relationships between tables instead of using primary/foreign key relationships. This has the affect of not only increasing

the number of joins required in order to obtain meaningful results, but also can make the transition to XML format more complex.

2. Data Access

As outlined above, the method for data access from the database needs to be determined. Most of the existing data propagation from the MIDB is provided via a series of database triggers and stored procedures, so this would be the expected method. A typical scenario might be that one or more updates occur to the data identified for XML transport, which fires a trigger that is in turn responsible for passing the data through the XML mapping and transformation engines.

One additional consideration is the method of interfacing with the data, which can take a number of different forms, and while the tools to be utilized may be affected by the method used, this will only be briefly discussed here since the XML transformations can be transparent to the data interface.

One interface method could involve ad hoc queries through an ODBC or JDBC interface. These queries could interface directly with the data or with stored procedures that implement the data queries. A number of the tools discussed in chapter IV provide APIs for this type of interface. While the mapping capabilities of these tools

typically can be used regardless of the data access method, certain advantages would result from using one of the standard interfaces.

Another interface method is to utilize existing application interfaces, such as those described in [GCC98], which provides an application interface to GCCS-I3 MIDB. Utilizing the published API would allow use of any intelligence (i.e. business rules) that is built into the API. Use of the API may also sometimes be a requirement due to security or other reasons, although with GCCS-I3 this is not the case. However, this is likely to greatly restrict the types of queries that can be processed, the result set that can be returned, and may in some cases negatively affect performance.

One method of publishing the data as XML is to use database *views*. Views allow data to be presented in a number of logical combinations that are independent of the underlying representation of the data. This is exactly what is needed in order to produce data as XML. By using a view to produce data structures that are the same as, or similar to, that of the targeted XML format, much of the data transformation has been done prior to extraction from the database. Combining views with a "publish as XML" feature in the database can allow virtually all of the transformation to XML to occur at the server side, reducing

the complexity and processing requirements of the client. This can also greatly improve performance at transaction time, since the actual transaction would not involve any of the overhead associated with data transformations and mapping.

Unfortunately, Sybase is not among the database vendors that currently support publishing views as XML. The use of views can be made, however, by publishing the subset of data that will be available for XML transactions. This will simplify the overall process and remove the overhead that would normally be required for performing joins and preparing the data for extraction.

3. XML Data Transformation Engine

One of the design steps addresses the need to transform data sets to one or more of the standard message formats. XSLT can be integrated here as part of the middleware solution for performing this translation. Both USMTF and CIX messaging formats currently have XML extensions. Applying XSLT with an appropriate stylesheet would allow the XML data to be transitioned to one of these formats.

Another type of transformation that could be made using XSLT would be to take the data to a display format. An example of this would be the ability to pull targeting data out of the MIDB for mission planning purposes, and format the data for display within a web browser.

Inserting an XQL processor in the XML data stream provides yet another method for filtering or transforming the XML data stream.

4. Method and Protocol for Data Transmission

An important consideration in the widely distributed DOD operating environment is how to distribute the data once it is in XML format. Two of the technologies discussed in Chapter IV would provide this capability, and both work in a similar manner.

Both the Simple Object Access Protocol (SOAP) and Microsoft's BizTalk framework have communication facilities for transmission and reception of XML data. They both perform message routing through the use of external wrappers on messages with XML content. These would be valid mechanisms for data transportation.

D. ASSESSMENT OF EFFECTIVENESS

It is possible to utilize XML and its associated technologies to greatly reduce the barriers to data interoperability. This issue is very important in the face of past failures in this area. It is clear from the previous assessment, however, that this is still not an easy task, there are many choices to be made, and the process is not risk free.

One conclusion that can be drawn about each of the products and technologies that have been reviewed, is that

they are relatively new to the market and they each represent the first generation in their respective categories. The result of this has been a lack of satisfactory case studies from which to draw conclusions. This fact, although it will certainly change over time, will increase risk for investments in products and technologies.

It is also evident from the study that there is no single product or technology that can be expected to accomplish each of the goals. Further, it appears that no single vendor seems to dominate in this area, and that each of the individual products work well together. The overall affect from this is an increased modularity within the architecture and decreased risk from not having a reliance on any single product or technology.

The most established set of products and technologies appear to be the middleware tools for performing data mapping and translations. This is true in terms of the number of products available and in their functionality. The XML-enabled DBMSs are mostly still in the early stages of adopting XML and, in the case of Sybase, appear to be lacking in functionality.

It is clear that some of the steps listed in Section B.4 will not be easily accomplished. Analyzing the existing legacy structures is never an easy task, even when using COTS products to assist with evaluations. Much of the

necessary information is buried in application code that can be very time consuming to analyze.

One of the challenges that must be overcome in translating between the extended messaging discussed here and standard message formats is handling the set of restrictions that standard messaging formats place on the structure of the messages. One example is the allowable line length and message length for the OTH-G format. OTH-G message lengths are limited to 100 lines, and line lengths are restricted to 69 characters [JWI00].

In surveying the COTS tools available on the open marketplace, it is clear that there are important distinctions that need to be made between the diverse set of requirements that exist within the DOD when it comes to data interoperability, and the more narrowly focused requirements presented by standard business applications. While many tools are being developed specifically to address data interchange between heterogeneous database systems, tools that are designed for use in the business to business arena are not always suited to handle some of the requirements that arise due to the size and diversity of data within the DOD.

One example of these differences is in the area of data access. While many of the COTS tools may require some standard method for interfacing with a database, such as

ODBC or JDBC, this may not be possible, or even allowed, when interfacing with certain DOD databases which require the use of a specific API for all data access.

E. CHAPTER SUMMARY

Although the tools and technologies do exist for providing shared data access from a legacy environment, it is clear that there is no general solution and that each system must be evaluated separately. It is also evident that the choices may not be clear and the maturity of the tools and the specifications on which they are based will be a big consideration.

Although the technology is still evolving, there is a great deal of work being done with XML, both within the DOD and in the business community. It will be important to look at parallel efforts in different fields. The Weather Observation Definition Format is one example of a parallel effort that can lend valuable insight.

VI. CONCLUSION AND FUTURE RESEARCH POSSIBILITIES

This thesis assesses some of the techniques and existing technologies for applying XML to the exchange of data between different database systems. While the standards and specific implementations that have been discussed represent the state of current technology, this work by no means represents the final chapter. XML and its associated standards are a moving target, with new uses and strategies for use being developed daily. This document can, however, provide a description of a process for the use of XML in data exchange and the problems that must be addressed.

The first part of this thesis identifies several of the issues affecting data interoperability within the DOD. The primary issue is that legacy systems, originally developed to address a single set of requirements, are very difficult and costly to modify to share data with other systems. At the data level, integrating differences in schemas and in the rules applied to those schemas is very difficult to achieve.

XML is well suited to help solve these issues. One of the primary driving factors behind the use of XML is that it provides the capability to develop common centralized schemas, without modifying the legacy database schemas.

This is critical since it addresses one of the main problems with past approaches to data integration.

Applying the use of XML is, by necessity, a phased approach. Recent developments have been directed at adapting standardized DOD messaging solutions to use XML. This provides immediate advantages, since operational systems that already rely on these messaging solutions can be easily adapted to utilize XML. This alone, however, will not provide data sharing to the extent required and it does not leverage the full advantage that can be provided by XML. The next step in the process involves detailed analysis of existing systems, the development of a common schema, and the application of data translations and mappings for each system.

The evaluation of XML tools and technologies and their application to a subset of the GCCS-I3 MIDB schema provided some insight into where problems exist and which questions remain unanswered. The primary observation that stood out, as might be expected with a relatively new technology, is that the tools are greatly mixed in terms of functionality and maturity. Another observation, which is really a characteristic of the XML design, is that multiple tools will probably need to be applied in order to accomplish the task.

An important question that remains to be answered is whether the application of available tools provides an adequate level of performance. This has been a criticism of XML in the past, and, although there are ways to improve the overall efficiency, this will require close evaluation. Another question that should be answered is what the timeframe would be for the design and implementation of a specific approach.

The next logical step is to apply some of the tools and technologies discussed to sharing a subset of data between two systems. This would initially take the form of a set of requirements and design specifications, followed by some prototype work. As suggested in this thesis, this work should take a step beyond the transmission and reception of USMTF or CIX messaging, although it would be good to utilize a filter to transition subsets of data to these message formats.

As specifications become solidified and the product base matures, new approaches to solving the data interoperability problem may surface. While it is highly unlikely that there exists a silver bullet approach that will solve all the problems, XML has the capability to greatly reduce costs and simplify efforts to create a viable data sharing solution.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX A (XML Document Listing)

```
<?xml version="1.0" encoding="UTF-8"?>
<Target_Mission>
     <Mission_ID>152-XX-221</Mission_ID>
     <Operation_Name>Tandem Thrust</Operation_Name>
     <Classification_Level>T</Classification_Level>
     <Codeword>5</Codeword>
     <Mission_Name>Strike Package 322</Mission_Name>
     <Target_Objective>
          <Country>IQ</Country>
          <Execution_Date>20000927</Execution_Date>
          <Functional_Production_Area>FUELS</Functional_Production_Area>
          <Priority_Objective>3</Priority_Objective>
          <Record_Status>E</Record_Status>
          <Domain_Level>SI</Domain_Level>
          <Eval>2</Eval>
          <Originating_Agency>EA</Originating_Agency>
          <Objective_Name>Airfield in Area 301</Objective_Name>
     </Target_Objective>
     <Target_List>
          <Operation_Name>Tandem Thrust</Operation_Name>
          <Classification_Level>T</Classification_Level>
          <Date_Created>20000825194500</Date_Created>
          <Date_Last_Changed>00000000000000</Date_Last_Changed>
          <Domain_Level>SI</Domain_Level>
          <Target_List_ID>12226</Target_List_ID>
          <Target_List_Status>A</Target_List_Status>
          <Target_List_Type>JTL</Target_List_Type>
          <Target_List_Name>Area 301 Tamino Airfield
Desig</Target_List_Name>
          <Production_Level>S</Production_Level>
          <Record_Status>E</Record_Status>
          <Target>
               <Affiliation>H</Affiliation>
               <Country>IQ</Country>
               <Classification_Level>T</Classification_Level>
               <Condition>COM</Condition>
               <Coordinates>325218290N1170928640E</Coordinates>
               <Coordinate_Basis>2</Coordinate_Basis>
               <Coordinate_Derivative>PM</Coordinate_Derivative>
               <Date_Created>19991011160000</Date_Created>
               <Date_Last_Change>00000000000000</Date_Last_Change>
               <Hardness>M</Hardness>
               <Height>320.0</Height>
               <Domain_Level>SI</Domain_Level>
               <Elevation>2040</Elevation>
               <Elevation_Confidence>100</Elevation_Confidence>
               <Target_Name>Control Tower</Target_Name>
               <Evaluation>1</Evaluation>
               <Radius>125.0</Radius>
               <Review_Date>20000825190000</Review_Date>
               <Release_Mark>MQ</Release_Mark>
               <Facility>
                    <Access>CLRMO</Access>
```

```
                    <Activity>ATC</Activity>
                    <BE_Number>1014-8Z-3967</BE_Number>
                    <Category>40812</Category>
                    <Evaluation>1</Evaluation>
                    <Facility_Name>Tamino Control Tower</Facility_Name>
                    <Facility_ID>32008</Facility_ID>
                    <Location_Name>Tamino Airfield</Location_Name>
                    <Primary_Mission>DQ</Primary_Mission>
                    <Relative_Ranking>1</Relative_Ranking>

<Population_Area_Proximity>14</Population_Area_Proximity>
                    <Record_Status>E</Record_Status>
                    <Review_Date>20000825190000</Review_Date>
                    <Graphic_Agency>DIA</Graphic_Agency>
                    <Graphic_Country>US</Graphic_Country>
               </Facility>
          </Target>
          <Target>
               <Affiliation>H</Affiliation>
               <Country>IQ</Country>
               <Classification_Level>T</Classification_Level>
               <Condition>COM</Condition>
               <Coordinates>325177560N1170823930E</Coordinates>
               <Coordinate_Basis>2</Coordinate_Basis>
               <Coordinate_Derivative>PM</Coordinate_Derivative>
               <Date_Created>19991011160000</Date_Created>
               <Date_Last_Change>00000000000000</Date_Last_Change>
               <Hardness>H</Hardness>
               <Height>20.0</Height>
               <Domain_Level>SI</Domain_Level>
               <Elevation>2040</Elevation>
               <Elevation_Confidence>100</Elevation_Confidence>
               <Target_Name>Bunker</Target_Name>
               <Evaluation>1</Evaluation>
               <Radius>235.0</Radius>
               <Review_Date>20000825190000</Review_Date>
               <Release_Mark>MQ</Release_Mark>
               <Facility>
                    <Access>CLRMO</Access>
                    <Activity>STG</Activity>
                    <BE_Number>1014-8Z-3976</BE_Number>
                    <Category>40812</Category>
                    <Evaluation>1</Evaluation>
                    <Facility_Name>Bunker for A/C Storage</Facility_Name>
                    <Facility_ID>32010</Facility_ID>
                    <Location_Name>Tamino Airfield</Location_Name>
                    <Primary_Mission>DQ</Primary_Mission>
                    <Relative_Ranking>2</Relative_Ranking>

<Population_Area_Proximity>14</Population_Area_Proximity>
                    <Record_Status>E</Record_Status>
                    <Review_Date>20000825190000</Review_Date>
                    <Graphic_Agency>DIA</Graphic_Agency>
                    <Graphic_Country>US</Graphic_Country>
               </Facility>
          </Target>
          <Target>
```

```
                <Affiliation>H</Affiliation>
                <Country>IQ</Country>
                <Classification_Level>T</Classification_Level>
                <Condition>COM</Condition>
                <Coordinates>325377680N1171033970E</Coordinates>
                <Coordinate_Basis>2</Coordinate_Basis>
                <Coordinate_Derivative>PM</Coordinate_Derivative>
                <Date_Created>19991011160000</Date_Created>
                <Date_Last_Change>19991205132000</Date_Last_Change>
                <Hardness>H</Hardness>
                <Height>0.0</Height>
                <Domain_Level>SI</Domain_Level>
                <Elevation>2040</Elevation>
                <Elevation_Confidence>100</Elevation_Confidence>
                <Target_Name>Runway</Target_Name>
                <Evaluation>1</Evaluation>
                <Radius>2000.0</Radius>
                <Review_Date>20000825190000</Review_Date>
                <Release_Mark>MQ</Release_Mark>
            </Target>
        </Target_List>
    </Target_Mission>
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX B (Sample Data DTD)

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT Access (#PCDATA)>
<!ATTLIST Activity
      dtype NMTOKEN #FIXED "string"
      dsize NMTOKEN #FIXED "3"
>
<!ELEMENT Activity (#PCDATA)>
<!ATTLIST Affiliation
      dtype NMTOKEN #FIXED "char"
>
<!ELEMENT Affiliation (#PCDATA)>
<!ATTLIST BE_Number
      dtype NMTOKEN #FIXED "string"
      dsize NMTOKEN #FIXED "10"
>
<!ELEMENT BE_Number (#PCDATA)>
<!ATTLIST Category
      dtype NMTOKEN #FIXED "string"
      dsize NMTOKEN #FIXED "5"
>
<!ELEMENT Category (#PCDATA)>
<!ATTLIST Classification_Level
      dtype NMTOKEN #FIXED "char"
>
<!ELEMENT Classification_Level (#PCDATA)>
<!ATTLIST Codeword
      dtype NMTOKEN #FIXED "char"
>
<!ELEMENT Codeword (#PCDATA)>
<!ATTLIST Condition
      dtype NMTOKEN #FIXED "string"
      dsize NMTOKEN #FIXED "4"
>
<!ELEMENT Condition (#PCDATA)>
<!ATTLIST Coordinate_Basis
      dtype NMTOKEN #FIXED "string"
      dsize NMTOKEN #FIXED "2"
>
<!ELEMENT Coordinate_Basis (#PCDATA)>
<!ATTLIST Coordinate_Derivative
      dtype NMTOKEN #FIXED "string"
      dsize NMTOKEN #FIXED "2"
>
<!ELEMENT Coordinate_Derivative (#PCDATA)>
<!ATTLIST Coordinates
      dtype NMTOKEN #FIXED "string"
      dsize NMTOKEN #FIXED "21"
>
<!ELEMENT Coordinates (#PCDATA)>
<!ATTLIST Country
      dtype NMTOKEN #FIXED "string"
      dsize NMTOKEN #FIXED "2"
>
```

```
<!ELEMENT Country (#PCDATA)>
<!ATTLIST Date_Created
      dtype NMTOKEN #FIXED "string"
      dsize NMTOKEN #FIXED "14"
>
<!ELEMENT Date_Created (#PCDATA)>
<!ATTLIST Date_Last_Change
      dtype NMTOKEN #FIXED "string"
      dsize NMTOKEN #FIXED "14"
>
<!ELEMENT Date_Last_Change (#PCDATA)>
<!ATTLIST Domain_Level
      dtype NMTOKEN #FIXED "string"
      dsize NMTOKEN #FIXED "2"
>
<!ELEMENT Domain_Level (#PCDATA)>
<!ATTLIST Elevation
      dtype NMTOKEN #FIXED "float"
>
<!ELEMENT Elevation (#PCDATA)>
<!ELEMENT Elevation_Confidence (#PCDATA)>
<!ATTLIST Eval
      dtype NMTOKEN #FIXED "char"
>
<!ELEMENT Eval (#PCDATA)>
<!ATTLIST Evaluation
      dtype NMTOKEN #FIXED "char"
>
<!ELEMENT Evaluation (#PCDATA)>
<!ATTLIST Execution_Date
      dtype NMTOKEN #FIXED "string"
      dsize NMTOKEN #FIXED "8"
>
<!ELEMENT Execution_Date (#PCDATA)>
<!ELEMENT Facility (Access, Activity, BE_Number, Category, Evaluation,
Facility_Name, Facility_ID, Location_Name?, Primary_Mission?,
Relative_Ranking?, Population_Area_Proximity?, Record_Status,
Review_Date, Graphic_Agency, Graphic_Country)>
<!ATTLIST Facility_ID
      dtype NMTOKEN #FIXED "string"
      dsize NMTOKEN #FIXED "14"
>
<!ELEMENT Facility_ID (#PCDATA)>
<!ATTLIST Facility_Name
      dtype NMTOKEN #FIXED "string"
      dsize NMTOKEN #FIXED "54"
>
<!ELEMENT Facility_Name (#PCDATA)>
<!ATTLIST Functional_Production_Area
      dtype NMTOKEN #FIXED "string"
      dsize NMTOKEN #FIXED "5"
>
<!ELEMENT Functional_Production_Area (#PCDATA)>
<!ATTLIST Graphic_Agency
      dtype NMTOKEN #FIXED "string"
      dsize NMTOKEN #FIXED "15"
>
```

```
<!ELEMENT Graphic_Agency (#PCDATA)>
<!ATTLIST Graphic_Country
      dtype NMTOKEN #FIXED "string"
      dsize NMTOKEN #FIXED "2"
>
<!ELEMENT Graphic_Country (#PCDATA)>
<!ATTLIST Hardness
      dtype NMTOKEN #FIXED "char"
>
<!ELEMENT Hardness (#PCDATA)>
<!ATTLIST Height
      dtype NMTOKEN #FIXED "float"
>
<!ELEMENT Height (#PCDATA)>
<!ATTLIST Location_Name
      dtype NMTOKEN #FIXED "string"
      dsize NMTOKEN #FIXED "54"
>
<!ELEMENT Location_Name (#PCDATA)>
<!ELEMENT Mission_ID (#PCDATA)>
<!ATTLIST Mission_ID
      dtype NMTOKEN #FIXED "string"
      dsize NMTOKEN #FIXED "15"
>
<!ATTLIST Mission_Name
      dtype NMTOKEN #FIXED "string"
      dsize NMTOKEN #FIXED "30"
>
<!ELEMENT Mission_Name (#PCDATA)>
<!ATTLIST Objective_Name
      dtype NMTOKEN #FIXED "string"
      dsize NMTOKEN #FIXED "54"
>
<!ELEMENT Objective_Name (#PCDATA)>
<!ATTLIST Operation_Name
      dtype NMTOKEN #FIXED "string"
      dsize NMTOKEN #FIXED "54"
>
<!ELEMENT Operation_Name (#PCDATA)>
<!ELEMENT Originating_Agency (#PCDATA)>
<!ATTLIST Population_Area_Proximity
      dtype NMTOKEN #FIXED "char"
>
<!ELEMENT Population_Area_Proximity (#PCDATA)>
<!ATTLIST Primary_Mission
      dtype NMTOKEN #FIXED "string"
      dsize NMTOKEN #FIXED "4"
>
<!ELEMENT Primary_Mission (#PCDATA)>
<!ATTLIST Priority_Objective
      dtype NMTOKEN #FIXED "smallint"
>
<!ELEMENT Priority_Objective (#PCDATA)>
<!ATTLIST Production_Level
      dtype NMTOKEN #FIXED "char"
>
<!ELEMENT Production_Level (#PCDATA)>
```

```
<!ATTLIST Radius
      dtype NMTOKEN #FIXED "float"
>
<!ELEMENT Radius (#PCDATA)>
<!ATTLIST Record_Status
      dtype NMTOKEN #FIXED "char"
>
<!ELEMENT Record_Status (#PCDATA)>
<!ATTLIST Relative_Ranking
      dtype NMTOKEN #FIXED "int"
>
<!ELEMENT Relative_Ranking (#PCDATA)>
<!ATTLIST Release_Mark
      dtype NMTOKEN #FIXED "string"
      dsize NMTOKEN #FIXED "2"
>
<!ELEMENT Release_Mark (#PCDATA)>
<!ATTLIST Review_Date
      dtype NMTOKEN #FIXED "string"
      dsize NMTOKEN #FIXED "14"
>
<!ELEMENT Review_Date (#PCDATA)>
<!ELEMENT Target (Affiliation?, Country?, Classification_Level,
Condition, Coordinates, Coordinate_Basis, Coordinate_Derivative,
Date_Created, Date_Last_Change, Hardness?, Height?, Domain_Level,
Elevation?, Elevation_Confidence, Target_Name, Evaluation, Radius?,
Review_Date, Release_Mark?, Facility?)>
<!ELEMENT Target_List (Operation_Name?, Classification_Level,
Date_Created, Date_Last_Change, Domain_Level, Target_List_ID,
Target_List_Status, Target_List_Type, Target_List_Name,
Production_Level, Record_Status, Target+)>
<!ATTLIST Target_List_ID
      dtype NMTOKEN #FIXED "string"
      dsize NMTOKEN #FIXED "14"
>
<!ELEMENT Target_List_ID (#PCDATA)>
<!ATTLIST Target_List_Name
      dtype NMTOKEN #FIXED "string"
      dsize NMTOKEN #FIXED "54"
>
<!ELEMENT Target_List_Name (#PCDATA)>
<!ATTLIST Target_List_Status
      dtype NMTOKEN #FIXED "string"
      dsize NMTOKEN #FIXED "3"
>
<!ELEMENT Target_List_Status (#PCDATA)>
<!ATTLIST Target_List_Type
      dtype NMTOKEN #FIXED "string"
      dsize NMTOKEN #FIXED "3"
>
<!ELEMENT Target_List_Type (#PCDATA)>
<!ELEMENT Target_Mission (Mission_ID?, Operation_Name?,
Classification_Level, Codeword?, Mission_Name, Target_Objective,
Target_List)>
<!ATTLIST Target_Name
      dtype NMTOKEN #FIXED "string"
      dsize NMTOKEN #FIXED "54"
```

```
>
<!ELEMENT Target_Name (#PCDATA)>
<!ELEMENT Target_Objective (Country?, Execution_Date?,
Functional_Production_Area?, Priority_Objective?, Record_Status,
Domain_Level, Eval, Originating_Agency, Objective_Name)>
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX C (Sample Data XDR)

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet href="http://schemas.biztalk.org/BizTalk/g9boxjl2.xsl"
type="text/xsl"?>
<Schema name="Targetlist-schema" xmlns="urn:schemas-microsoft-com:xml-
data" xmlns:dt="urn:schemas-microsoft-com:datatypes">
     <ElementType name="Access" model="closed" content="textOnly"
dt:type="string" dt:maxLength="9"/>
     <ElementType name="Activity" model="closed" content="textOnly"
dt:type="string" dt:maxLength="3"/>
     <ElementType name="Affiliation" model="closed" content="textOnly"
dt:type="char"/>
     <ElementType name="BE_Number" model="closed" content="textOnly"
dt:type="string" dt:maxLength="10"/>
     <ElementType name="Category" model="closed" content="textOnly"
dt:type="string" dt:maxLength="5"/>
     <ElementType name="Classification_Level" model="closed"
content="textOnly" dt:type="char"/>
     <ElementType name="Codeword" model="closed" content="textOnly"
dt:type="char"/>
     <ElementType name="Condition" model="closed" content="textOnly"
dt:type="string" dt:maxLength="4"/>
     <ElementType name="Coordinate_Basis" model="closed"
content="textOnly" dt:type="string" dt:maxLength="2"/>
     <ElementType name="Coordinate_Derivative" model="closed"
content="textOnly" dt:type="string" dt:maxLength="2"/>
     <ElementType name="Coordinates" model="closed" content="textOnly"
dt:type="string" dt:maxLength="21"/>
     <ElementType name="Country" model="closed" content="textOnly"
dt:type="string" dt:maxLength="2"/>
     <ElementType name="Date_Created" model="closed" content="textOnly"
dt:type="string" dt:maxLength="14"/>
     <ElementType name="Date_Last_Change" model="closed"
content="textOnly" dt:type="string" dt:maxLength="14"/>
     <ElementType name="Date_Last_Changed" model="closed"
content="textOnly" dt:type="string" dt:maxLength="14"/>
     <ElementType name="Domain_Level" model="closed" content="textOnly"
dt:type="string" dt:maxLength="2"/>
     <ElementType name="Elevation" model="closed" content="textOnly"
dt:type="float"/>
     <ElementType name="Elevation_Confidence" model="closed"
content="textOnly" dt:type="i1"/>
     <ElementType name="Eval" model="closed" content="textOnly"
dt:type="char"/>
     <ElementType name="Evaluation" model="closed" content="textOnly"
dt:type="char"/>
     <ElementType name="Execution_Date" model="closed"
content="textOnly" dt:type="string" dt:maxLength="14"/>
     <ElementType name="Facility" model="closed" content="eltOnly"
order="seq">
          <element type="Access" minOccurs="0" maxOccurs="1"/>
          <element type="Activity" minOccurs="1" maxOccurs="1"/>
          <element type="BE_Number" minOccurs="1" maxOccurs="1"/>
          <element type="Category" minOccurs="1" maxOccurs="1"/>
```

```
            <element type="Evaluation" minOccurs="1" maxOccurs="1"/>
            <element type="Facility_Name" minOccurs="1" maxOccurs="1"/>
            <element type="Facility_ID" minOccurs="1" maxOccurs="1"/>
            <element type="Location_Name" minOccurs="0" maxOccurs="1"/>
            <element type="Primary_Mission" minOccurs="0" maxOccurs="1"/>
            <element type="Relative_Ranking" minOccurs="0" maxOccurs="1"/>
            <element type="Population_Area_Proximity" minOccurs="0"
maxOccurs="1"/>
            <element type="Record_Status" minOccurs="1" maxOccurs="1"/>
            <element type="Review_Date" minOccurs="1" maxOccurs="1"/>
            <element type="Graphic_Agency" minOccurs="1" maxOccurs="1"/>
            <element type="Graphic_Country" minOccurs="1" maxOccurs="1"/>
        </ElementType>
        <ElementType name="Facility_ID" model="closed" content="textOnly"
dt:type="string" dt:maxLength="14"/>
        <ElementType name="Facility_Name" model="closed" content="textOnly"
dt:type="string" dt:maxLength="54"/>
        <ElementType name="Functional_Production_Area" model="closed"
content="textOnly" dt:type="string" dt:maxLength="5"/>
        <ElementType name="Graphic_Agency" model="closed"
content="textOnly" dt:type="string" dt:maxLength="15"/>
        <ElementType name="Graphic_Country" model="closed"
content="textOnly" dt:type="string" dt:maxLength="2"/>
        <ElementType name="Hardness" model="closed" content="textOnly"
dt:type="char"/>
        <ElementType name="Height" model="closed" content="textOnly"
dt:type="float"/>
        <ElementType name="Location_Name" model="closed" content="textOnly"
dt:type="string" dt:maxLength="54"/>
        <ElementType name="Mission_ID" model="closed" content="textOnly"
dt:type="string" dt:maxLength="15"/>
        <ElementType name="Mission_Name" model="closed" content="textOnly"
dt:type="string" dt:maxLength="54"/>
        <ElementType name="Objective_Name" model="closed"
content="textOnly" dt:type="string" dt:maxLength="54"/>
        <ElementType name="Operation_Name" model="closed"
content="textOnly" dt:type="string" dt:maxLength="54"/>
        <ElementType name="Originating_Agency" model="closed"
content="textOnly" dt:type="string" dt:maxLength="2"/>
        <ElementType name="Population_Area_Proximity" model="closed"
content="textOnly" dt:type="char"/>
        <ElementType name="Primary_Mission" model="closed"
content="textOnly" dt:type="string" dt:maxLength="4"/>
        <ElementType name="Priority_Objective" model="closed"
content="textOnly" dt:type="i1"/>
        <ElementType name="Production_Level" model="closed"
content="textOnly" dt:type="char"/>
        <ElementType name="Radius" model="closed" content="textOnly"
dt:type="float"/>
        <ElementType name="Record_Status" model="closed" content="textOnly"
dt:type="char"/>
        <ElementType name="Relative_Ranking" model="closed"
content="textOnly" dt:type="i1"/>
        <ElementType name="Release_Mark" model="closed" content="textOnly"
dt:type="string" dt:maxLength="2"/>
        <ElementType name="Review_Date" model="closed" content="textOnly"
dt:type="string" dt:maxLength="14"/>
```

```
     <ElementType name="Target" model="closed" content="eltOnly"
 order="seq">
          <element type="Affiliation" minOccurs="0" maxOccurs="1"/>
          <element type="Country" minOccurs="0" maxOccurs="1"/>
          <element type="Classification_Level" minOccurs="1"
maxOccurs="1"/>
          <element type="Condition" minOccurs="1" maxOccurs="1"/>
          <element type="Coordinates" minOccurs="1" maxOccurs="1"/>
          <element type="Coordinate_Basis" minOccurs="1" maxOccurs="1"/>
          <element type="Coordinate_Derivative" minOccurs="1"
maxOccurs="1"/>
          <element type="Date_Created" minOccurs="1" maxOccurs="1"/>
          <element type="Date_Last_Change" minOccurs="1" maxOccurs="1"/>
          <element type="Hardness" minOccurs="0" maxOccurs="1"/>
          <element type="Height" minOccurs="0" maxOccurs="1"/>
          <element type="Domain_Level" minOccurs="1" maxOccurs="1"/>
          <element type="Elevation" minOccurs="0" maxOccurs="1"/>
          <element type="Elevation_Confidence" minOccurs="0"
maxOccurs="1"/>
          <element type="Target_Name" minOccurs="1" maxOccurs="1"/>
          <element type="Evaluation" minOccurs="1" maxOccurs="1"/>
          <element type="Radius" minOccurs="0" maxOccurs="1"/>
          <element type="Review_Date" minOccurs="1" maxOccurs="1"/>
          <element type="Release_Mark" minOccurs="0" maxOccurs="1"/>
          <element type="Facility" minOccurs="0" maxOccurs="1"/>
     </ElementType>
     <ElementType name="Target_List" model="closed" content="eltOnly"
order="seq">
          <element type="Operation_Name" minOccurs="0" maxOccurs="1"/>
          <element type="Classification_Level" minOccurs="1"
maxOccurs="1"/>
          <element type="Date_Created" minOccurs="1" maxOccurs="1"/>
          <element type="Date_Last_Changed" minOccurs="1" maxOccurs="1"/>
          <element type="Domain_Level" minOccurs="1" maxOccurs="1"/>
          <element type="Target_List_ID" minOccurs="1" maxOccurs="1"/>
          <element type="Target_List_Status" minOccurs="1"
maxOccurs="1"/>
          <element type="Target_List_Type" minOccurs="1" maxOccurs="1"/>
          <element type="Target_List_Name" minOccurs="1" maxOccurs="1"/>
          <element type="Production_Level" minOccurs="1" maxOccurs="1"/>
          <element type="Record_Status" minOccurs="1" maxOccurs="1"/>
          <element type="Target" minOccurs="1" maxOccurs="*"/>
     </ElementType>
     <ElementType name="Target_List_ID" model="closed"
content="textOnly" dt:type="string" dt:maxLength="14"/>
     <ElementType name="Target_List_Name" model="closed"
content="textOnly" dt:type="string" dt:maxLength="54"/>
     <ElementType name="Target_List_Status" model="closed"
content="textOnly" dt:type="string" dt:maxLength="3"/>
     <ElementType name="Target_List_Type" model="closed"
content="textOnly" dt:type="string" dt:maxLength="3"/>
     <ElementType name="Target_Mission" model="closed" content="eltOnly"
order="seq">
          <AttributeType name="xmlns" dt:type="string"/>
          <attribute type="xmlns"/>
          <element type="Mission_ID" minOccurs="0" maxOccurs="1"/>
          <element type="Operation_Name" minOccurs="0" maxOccurs="1"/>
```

```
          <element type="Classification_Level" minOccurs="1"
maxOccurs="1"/>
          <element type="Codeword" minOccurs="0" maxOccurs="1"/>
          <element type="Mission_Name" minOccurs="0" maxOccurs="1"/>
          <element type="Target_Objective" minOccurs="1" maxOccurs="*"/>
          <element type="Target_List" minOccurs="1" maxOccurs="*"/>
     </ElementType>
     <ElementType name="Target_Name" model="closed" content="textOnly"
dt:type="string" dt:maxLength="54"/>
     <ElementType name="Target_Objective" model="closed"
content="eltOnly" order="seq">
          <element type="Country" minOccurs="0" maxOccurs="1"/>
          <element type="Execution_Date" minOccurs="0" maxOccurs="1"/>
          <element type="Functional_Production_Area" minOccurs="0"
maxOccurs="1"/>
          <element type="Priority_Objective" minOccurs="0"
maxOccurs="1"/>
          <element type="Record_Status" minOccurs="1" maxOccurs="1"/>
          <element type="Domain_Level" minOccurs="1" maxOccurs="1"/>
          <element type="Eval" minOccurs="1" maxOccurs="1"/>
          <element type="Originating_Agency" minOccurs="0"
maxOccurs="1"/>
          <element type="Objective_Name" minOccurs="1" maxOccurs="1"/>
     </ElementType>
</Schema>
```

LIST OF REFERENCES

[ABI00] Abiteboul, S., Buneman, P., and Suciu, D., *Data on the Web: From Relations to Semistructured Data and XML.* Morgan Kaufmann Publishers, 2000.

[BOS99] Bosworth, A., Layman, A.,and Rys, M., "Serializing Graphs of Data in XML," http://www.biztalk.org/Resources/canonical.asp, Microsoft Corporation, 1999.

[BOU99] Bourret, R., "XML and Databases," Technical University of Darmstadt, http://www.informatik.tu-darmstadt.de/DVS1/staff/bourret/xml/XMLAndDatabases.htm, September 1999.

[BOX00] Box, D., et al., "SOAP: Simple Object Access Protocol," http://msdn.microsoft.com/xml/general/soapspec.asp, March, 2000.

[BRA98] Bray, T., "The Annotated XML Specification," http://www.xml.com/axml/axml.html, September 1998.

[BUC00] Buck, L., "Modeling Relational Data in XML," Extensibility, Inc., http://www.extensibility.com/xml_resources/modeling.htm, March 2000.

[CAR00] Carnevale, R., "The Joint Common Database," February 2000.

[DAT00] IBM, "Datajoiner," document available online at http://www.software.ibm.com/data/datajoiner.

[DAV99] David, M., "SQL-based XML Structured Data Access," Web Techniques, San Francisco, June 1999.

[DOM00] Wood, L., et al., "Document Object Model (DOM) Level 1 Specification," W3C Working Draft, http://www.w3.org/TR/2000/WD-DOM-Level-1-20000929/, September 2000.

[DTS00] Microsoft, "Data Transformation Services Considerations," document available online at http://msdn.microsoft.com/library/psdk/sql/dts_adv_15.htm

[FGM98] "Functional Description Document: Track and Relational Data Synchronization", Version 1.0, FGM Inc., August 1998.

[FLO99] Florescu, D. and Kossmann, D., "A Performance Evaluation of Alternative Mapping Schemes for Storing XML Data in a Relational Database," The French National Institute for Research in Computer Science and Control, May 1999.

[GCC98] "Global Command and Control System (GCCS) Application Program Interface Reference Manual (APIRM) for NIPS Developers Segment (NDEV)," Version 3.1.1.0, prepared for the Defense Information Systems Agency, April 1998.

[HAS00] Hayes, G., Pipher, J., and Davis, S., "Joint Common Catalog (JCC): Refined Concept and Implementation Status," *Proceedings for the Federal Database Colloquium and Exposition*, September 2000.

[HOD00] Hodges, A. and Buck, T., "Defense Information Infrastructure (DII) Common Operating Environment (COE) Data Access Services," *Proceedings of the Federal Database Colloquium and Exposition*, September 2000.

[INR00] "Implementation Guidelines for Interoperability with the GCCS-COP for JWID 2000", Inter-National Research Institute, Inc., May 2000.

[ISO86] "Information Processing – Text and Office Systems – Standard Generalized Markup Language (SGML)," ISO 8879, 1986.

[ISO97] "Framework for the Specification and Standardization of Data Elements," http://pueblo.lbl.gov/~olken/X3L8/drafts/draft.docs.html, 1998.

[JBM00] Myers, R. and Brunn, B., "Joint Battle Management Initiative (JBMI) Assessment Plan Draft," August 2000.

[JDB00] JavaSoft, "JDBC Data Access API," document available online at http://www.javasoft.com/products/jdbc/index.html.

[LAY98] Layman, A. et al., "XML-Data," World Wide Web Consortium (W3C) Note, http://www.w3.org/TR/1998/NOTE-XML-data-0105/, January 1998.

[MID98] "Modernized Integrated Database: Database Design Document," Defense Intelligence Agency, June 1998.

[MTF00] "XML-MTF Mapping, Third Public Working Draft," XML-MTF Development Team, August 2000.

[NRC99] "Realizing the Potential of C4I: Fundamental Challenges," Committee to Review DOD C4I Plans and Programs, National Research Council, 1999.

[ODB95] Microsoft, "ODBC Related Standards", http://ourworld.compuserve.com/homepages/VBrant/stds.htm, 1995.

[OMF00] "Weather Observation Definition Format (OMF)," http://zowie.metnet.navy.mil/~spawar/JMV-TNG/XML/OMF.html, Space and Naval Warfare PMW-185, May 2000.

[QUI00] Quin, L., *Open Source XML Database Toolkit: Resources and Techniques for Improved Development*, Wiley Computer Publishing, August 2000.

[REN96] Renner, S. and Scarano, J., "Migrating Legacy Applications to a Shared Data Environment," *Proceedings of the Federal Database Colloquium and Exposition*, August 1996.

[ROS00] Rosenthal, A., Frank, M., and Renner, S., "Getting Data to Applications: Why We Fail, How We Can Do Better," *Proceedings of the Federal Database Colloquium and Exposition 2000*, September 2000.

[ROS97] Rosenthal, A., Edward, S., and Renner, S., "Toward Unified Metadata for the Department of Defense," *IEEE Metadata Workshop*, April 1997.

[SCH00] Schneider, J., Cherinka, R., Cokus, M., and Molloy, M., "TBMCS-JBI Using XML-MTF Information Objects: An XML Technology Report for the Joint Battle Management Integration (JBMI) Assessment," The Mitre Corporation, January 2000.

[SYA00] Sybase Inc., "Adaptive Server Enterprise 12.0," document available online at http://www.sybase.com/products/databaseservers/ase/.

[SYB00] "Using XML with the Sybase Adaptive Server SQL Database," Technical Whitepaper, Sybase Inc., January 2000.

[WAP99] "WAP Binary XML Content Format," http://www1.wapforum.org/tech/documents/SPEC-WBXML-19991104.pdf, November 1999.

[XDB00] Bourret, R., "XML-DBMS," document available online at http://www.rpbourret.com/xmldbms/index.htm.

[XML98] "Extensible Markup Language 1.0," W3C Recommendation, http://www.w3.org/TR/REC-xml, February 1998.

[XQL00] "XML Query Requirements," W3C Working Draft, http://www.w3.org/TR/xmlquery-req, August 2000.

[XQL98] Ishikawa, H., Kubota, K., Kanemasa, Y., "XQL: A Query Language for XML Data," http://www.w3.org/TandS/QL/QL98/pp/flab.txt, November 1998.

[XSL99] "XSL Transformations," World Wide Web Consortium (W3C), http://www.w3.org/TR/xslt, November 1999.

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center 2
   8725 John J. Kingman Road, Ste 0944
   Ft. Belvoir, VA 22060-6218

2. Dudley Knox Library 2
   Naval Postgraduate School
   411 Dyer Rd.
   Monterey, CA 93943-5101

3. Chairman, Department of Computer Science 1
   Code CS
   Naval Postgraduate School
   Monterey, CA 93943

4. Software Engineering Program 1
   Computer Science Department
   Naval Postgraduate School
   Monterey, CA 93943

5. Space and Naval Warfare Systems Center, D40 1
   5356 Hull Street
   San Diego, CA 92152-5001

6. Space and Naval Warfare Systems Center, Code D42 1
   5356 Hull Street
   San Diego, CA 92152-5001

7. SSC SD Technical Library, Code D0274 1
   5356 Hull Street
   San Diego, CA 92152-5001

8. Dr. Valdis Berzins, Code CS/Bz 1
   Computer Science Department
   Naval Postgraduate School
   Monterey, CA 93943

9. Prof. Luqi, Code CS/Lq 1
   Computer Science Department
   Naval Postgraduate School
   Monterey, CA 93943

10. Capt. Paul Young 1
    Computer Science Department
    Naval Postgraduate School
    Monterey, CA 93943

11. Gerard Hufstetler..... 1
    Joint C4ISR Battle Center
    116 Lake View Parkway
    Suite 150
    Suffolk, VA 23435

12. CDR Jim Ward..... 1
    Joint C4ISR Battle Center
    116 Lake View Parkway
    Suite 150
    Suffolk, VA 23435

13. Dave Hina, Code D4223 3
    Space and Naval Warfare Systems Center
    53560 Hull Street
    San Diego, CA 92152-5001